Thought Note

Here, I am creating 22 pages html with Bootstrap and 5 JavaScript File. All the html and JavaScript file are given Below: -

Note:- This Project must be run on live server because I used the header, footer and product dynamically using JavaScript.

HTML File	JavaScript File
1. Index.html	1. Index.js
2. Login.html	2. Function.js
3. Contact.html	3. Routes.js
4. Cart.html	4. Mainelement.js
5. Header.html	5. Storage.js
6. Footer.html	
7. Product.html	
8. All.html (Kids)	
9. All.html (Men)	
10. All.html (Women)	
11. Bottom.html	
12. Top.html	
13. Inner.html	
14. Onepc.html	
15. Shirts.html	
16. Pants.html (Men)	
17. Trousers.html	
18. Jeans.html	
19. Dresses.html	
20. Pants.html (Women)	
21. Skirts.html	
22. Gowns.html	

Structure of Index.js while creating this file

1. **Import Statements**: The code imports specific functions (**addheaddata**, **createElement**, and **loadcontent**) from the **function.js** file using ES6 module syntax.
2. **DOM Content Load Event Listener**: An event listener is added for the DOMContentLoaded event, which fires when the initial HTML document

has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.

3. **Document Ready Function**: Inside the DOMContentLoaded event listener, several actions are performed:
    - The **addheaddata** function is called to add head data.
    - Header, main, and footer elements are created using the **createElement** function.
    - Content for the header, main (based on the current pathname), and footer is loaded asynchronously using the **loadcontent** function.
4. **Pathname Variable**: The **pathname** variable stores the current URL pathname using **window.location.pathname**, which represents the path component of the URL.

Structure of Function.js while creating this file

1. **createElement**: This function creates an HTML element of a specified type, assigns it an ID based on the type, and appends it to the document body.
2. **addheaddata**: This function dynamically adds meta tags, title, favicon, stylesheets, and scripts to the document head.
3. **loadcontent**: This function loads content from a specified URL using the Fetch API, inserts it into a specified element, and optionally changes the URL using the History API. It also triggers routing and main element creation based on the target ID.
4. **createproductcard**: This private function creates a product card HTML element based on the provided product data.
5. **addproduct**: This function adds product cards to the specified product list elements based on the provided item array.
6. **logincheck**: This function performs login validation by checking the values of email and password fields. It alerts the user with a success message if the credentials match, otherwise alerts with an error message.

Structure of Routes.js while creating this file

1. **addclickelements**: This function takes an array of elements and a page URL as input. It attaches click event listeners to each element in the array. When clicked, it calls the **loadcontent** function to load content into the main element, passing 'Y' to indicate that the URL should be changed.
2. **routing**: This function defines routes for various elements based on their IDs and maps them to specific page URLs. It iterates through each route, selects the corresponding elements using **document.querySelectorAll**, and attaches click event listeners to them using **addclickelements**.

Structure of Mainelement.js while creating this file

1. **Product Mapping**: It defines a mapping between specific URLs and corresponding product lists to be loaded into specific elements on the page.
2. **Product Loading**: It iterates through the product lists defined for the current URL and adds products to the corresponding elements using the **addproduct** function from the **function.js** module.
3. **Switch Statement**: It contains a switch statement that handles different cases based on the URL. Depending on the URL, it loads different product lists into specific elements on the page.
4. **Login Page Handling**: It adds an event listener to the login button to perform login validation when clicked.
5. **Index Page Handling**: It loads the general product list onto the index page.
6. **Product Page Handling**: It loads specific product lists onto the product page based on the URL parameters.

Structure of Index.html while creating this file

1. **Head Section**: This contains metadata and links to external resources like stylesheets and scripts. In this case, it's importing a JavaScript file called

"index.js" using the **defer** attribute, which means it will be executed after the DOM is fully parsed.

2. **Body Section**: This is where the visible content of the webpage resides.
    - **Main Container**: The ** ** element with the id "container" wraps the entire content of the webpage.
    - **Section with Background Image**: This section contains a **<figure>** element with a background image, likely serving as a hero section with a logo and slogan overlaid on top of the image.
    - **Section for Featured Products**: This section is for displaying featured products. It contains a heading and a carousel (** **) for cycling through multiple product items.
        - **Carousel Items**: Each carousel item (** **) contains a row (** **) where product content will be dynamically loaded.
        - **Carousel Control Buttons**: These are buttons for navigating the carousel. There's a "Previous" button and a "Next" button.

Structure of Login.html while creating this file

1. **Script Tag**: It imports a JavaScript file named "index.js" from a parent directory. The **defer** attribute ensures that the script is executed after the DOM is fully parsed.
2. **Container**: This ** ** with the id "container" contains the entire login form. It has classes for styling purposes (**container-fluid**, **p-4**, **my-5**).
3. **Form**: The **<form>** element with the id "loginform" is the actual login form. It's set to submit data to "#" (meaning the current page) using the HTTP POST method.
4. **Legend**: The **<legend>** element provides a title for the form, indicating that it's a login form. It's styled as a large, bold text in the center of the container.

5. **Email Input Field**: This section contains an input field for the user's email address. It has a label, an **<input>** field of type "email", and a placeholder text.
6. **Password Input Field**: Similar to the email input, this section contains an input field for the user's password. It has a label, an **<input>** field of type "password", and a placeholder text.
7. **Login Button**: This is an **<input>** element of type "button" with the id "login". It serves as the submit button for the form. When clicked, it triggers the login process.

Structure of Contact.html while creating this file

1. **Script Tag**: Similar to the previous examples, it includes a JavaScript file named "index.js" to handle dynamic functionalities, presumably for the header and footer.
2. **Container for the Contact Form**: This ** ** with the id "container" holds the entire content of the contact page. It has a class for styling purposes (**container**, **text-center**, **my-4**).
3. **Heading**: The **<h1>** element displays the title "Contact Us" at the top of the page.
4. **Row with Two Columns**: This ** ** contains two columns. One column holds an image related to contacting, while the other contains the contact form.
    - **Column for Image**: This column (** **) displays an image related to contacting. It's responsive and fills the entire height of the column.
    - **Column for Form**: This column (** **) contains the contact form. It's styled to be left-aligned and has some padding.
5. **Contact Form**: The **<form>** element is where users input their contact information and message.
    - **Name, Email, Phone Number, Message Input Fields**: Each input field (**<input>** or **<textarea>**) is accompanied by a label and a placeholder. These fields collect the user's name, email, phone number, and message respectively.

- **Submit Button**: The submit button (**<input type="submit">**) allows users to send their message. It's styled with a background color, border, and rounded corners.

Structure of Cart.html while creating this file

1. **Script Tag**: It includes a JavaScript file named "index.js" to handle dynamic functionalities for the header and footer. The **defer** attribute ensures that the script is executed after the DOM is fully parsed.
2. **Container for Cart Items**: This ** ** with the id "container" holds the content of the shopping cart page. It has classes for styling purposes (**container-fluid**, **container-xxl**).
3. **Row with Columns**: Inside the container, there's a ** ** with the class "row" containing two columns.
    - **Column for Cart Item Details**: The first column (** **) contains the details of items in the cart, displayed within a card. Each cart item is represented by a row containing columns for item image, details, and quantity adjustment buttons.
        - **Item Image Column**: Displays the image of the item.
        - **Item Details Column**: Displays the name of the item and buttons to remove the item from the cart or move it to the wishlist.
        - **Item Quantity Column**: Allows users to adjust the quantity of the item in the cart using plus and minus buttons.
    - **Column for Cart Summary**: The second column (** **) contains a summary of the items in the cart, displayed within a card. It includes the total cost, shipping cost, and the overall total. Additionally, there's a checkout button at the bottom of the summary

Structure of Header.html while creating this file

1. **Style Tag**: This contains CSS for a dropdown menu to display on hover. The **.dropdown:hover > .dropdown-menu** selector ensures that the dropdown menu appears when hovering over the dropdown link.

2. **Navbar**: The ** ** creates the navigation bar. It consists of the logo, a toggle button for collapsing on smaller screens, and a search bar. On larger screens, it also includes links for login and cart.
3. **Navigation Links**: The **<nav class="nav nav-pills justify-content-center bg-body-secondary bg-opacity-75 px-4 py-0">** section contains navigation links for different pages/categories.
    - **Home and All Products Links**: These are simple links for the home page and all products page.
    - **Dropdown Menus**: There are dropdown menus for Women, Men, and Kids categories. Each dropdown menu contains links to specific subcategories.
4. **Dropdown Menus**: Each dropdown menu is structured with a list of links inside a **<ul class="dropdown-menu fs-5 fw-bold">**. The links are structured as list items (**<li>**) with anchor tags (**<a>**). The **href** attribute of these anchor tags can be updated to link to the respective pages or sections.

Structure of Footer.html while creating this file

1. **Footer Section**: The ** ** sets up the footer with a dark background color and padding.
2. **Row with Columns**: Inside the footer, there's a ** ** containing columns for different sections.
    - **Columns for Women's, Men's, and Kids' Sections**: Each column (** **) contains links to specific clothing categories for women, men, and kids respectively. The links are styled as a list (**<ul>**) with list items (**<li>**).
    - **Column for General Links**: This column contains general links such as Home, Login, Contact, and Cart.
3. **Horizontal Line Separator**: The **<hr class="text-light m-0" />** creates a horizontal line separator with light color.
4. **Copyright Statement**: The **<p class="text-center fs-5 py-4 my-0">** displays a copyright statement with the year and the website name.

Structure of Product.html while creating this file

1. **Script Tag**: This includes a JavaScript file named "index.js" to handle dynamic functionalities. The **defer** attribute ensures that the script is executed after the DOM is fully parsed.
2. **Main Container**: The ** ** creates the main container with Bootstrap classes for responsive design.
3. **Sections for Women, Men, and Kids**:
    - **Women Section**: This section (** **) is for displaying women's clothing items. It includes a heading **<h2>** with a background color and rounded corners. The content for women's items will be dynamically loaded inside this section.
    - **Men Section**: Similar to the women's section, this section (** **) is for displaying men's clothing items. It also has a heading **<h2>** with a background color and rounded corners, and the content will be dynamically loaded here.
    - **Kids Section**: This section (** **) is for displaying kids' clothing items. It follows the same structure as the women's and men's sections, with a heading **<h2>** and dynamically loaded content.

Structure of all.html for Men, Women and Kids while creating this file

1. **Script Tag**: This includes a JavaScript file named "index.js" to handle dynamic functionalities. The **defer** attribute ensures that the script is executed after the DOM is fully parsed.
2. **Main Container**: The ** ** creates the main container with Bootstrap classes for responsive design.

Structure of Separate Category of each html while creating this file

1. **Script Tag**: This includes a JavaScript file named "index.js" to handle dynamic functionalities. The **defer** attribute ensures that the script is executed after the DOM is fully parsed.
2. **Main Container**: The ** ** creates the main container with Bootstrap classes for responsive design.
3. **Title for Category**: This section includes an **<h2>** element with the title of Category. It has classes for styling such as margin, padding, background color, and rounded corners.
4. **Row for Displaying Products**: This ** ** is designated for displaying products. It will contain dynamically loaded product items related to Category.