Informatics Institute of Technology
Department of Computing

BSc (Hons) Artificial Intelligence and Data Science

Module: CM1605 Web Technology

Module Coordinator: Ms. Janani Harischandra

Group Coursework Report

Tutorial Group : B

Student Details :

Coursework Group No : Group 2

Students:

RGU ID	IIT Student ID	Student Name
2313081	20221470	W.L.A. Cooray
2313473	20221151	Y.M. Karunanayake
2313075	20220494	Mu-aadh Nazly
2313319	20221081	B. P. M. Peiris

Contents

1. Introduction

This report demonstrates the development of an interactive website called Chapter, which is a book recommendation system. This system provides users with the ability to create personalized profiles, a gallery page to view the pictures and details of the books, a favorite list to create a personalized library of past-read books, and related items. Furthermore, the shopping cart allows the user to buy products that are related to books, and there is a quiz to improve their knowledge about books. The names of the group-members are:

- Yasini Mandara Karunanayake RGU-ID: 2313473 – Student-Role-1
- Muhammadh Nazly Muaadh RGU-ID: 2313075 – Student-Role-2
- Lakshan Anjana Cooray RGU-ID: 2313081 – Student-Role-3
- Praveen Madhawa Peiris RGU-ID: 2313319 – Student-Role-4

2. Research on Existing System

1. W3schools

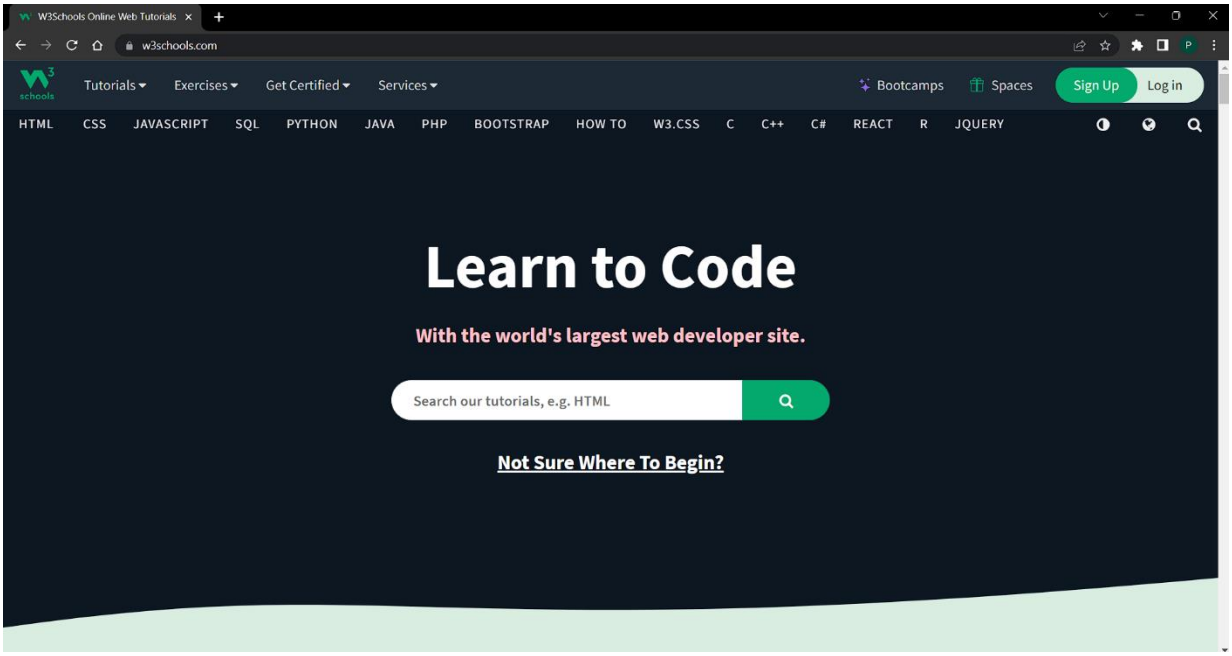


Figure 0.1 W3schools homepage (w3schools.com)

W3Schools is a valuable learning tool for web development, catering to developers and students interested in HTML, CSS, JavaScript, and related technologies. The site's accessible and user-friendly layout allows easy navigation through its well-organized structure, featuring a prominent search box and clear menu. With consistent typefaces and color schemes, the website ensures a visually appealing experience.

Its step-by-step resources enable efficient comprehension of concepts, making it ideal for beginners and experienced developers alike. Complex topics become less overwhelming, as students can gradually build their coding skills through tutorials. Overall, W3Schools' features and design establish it as a crucial instructional resource, supporting its position in the web development community. It is a suitable platform for students at all levels of expertise.

2. Books wagon

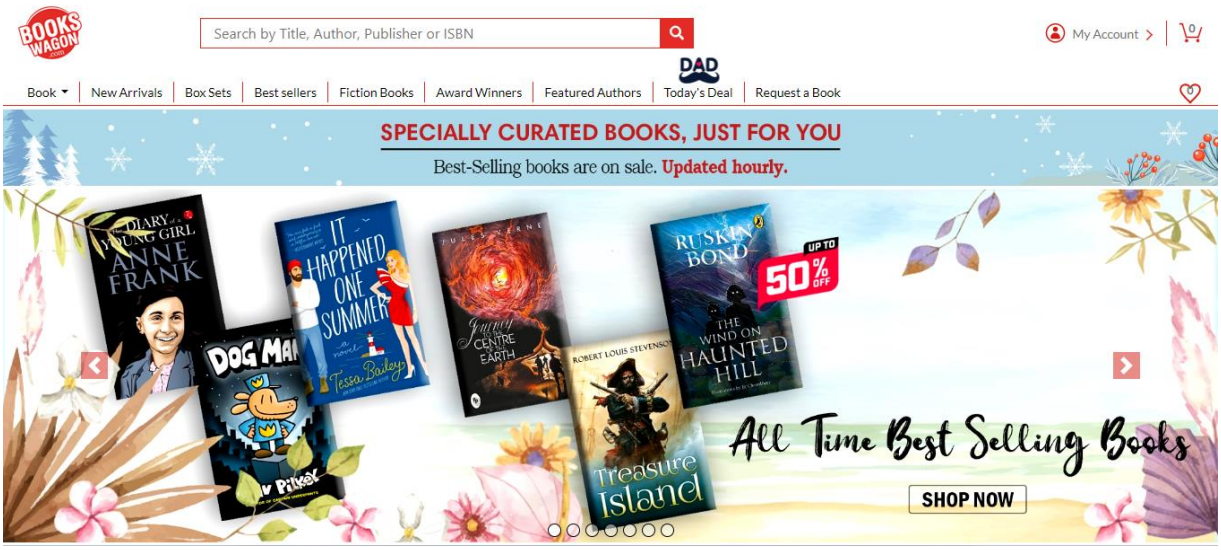


Figure 0.2 Books wagon homepage (www.bookswagon.com)

Books Wagon is an online bookstore offering a wide range of books in various categories, with over 40 million selections. The website's organized and user-friendly UI features highlighted sections for featured books, bestsellers, and new releases, enhancing the user experience. A search bar at the top enables quick access to books, authors, or genres.

Thanks to its responsive design, Books Wagon provides a consistent browsing experience on PCs, tablets, and smartphones. Users can sign up and purchase books based on their preferences, benefiting from great deals and discounts. The website also suggests related books and latest releases, simplifying the discovery of new titles.

The responsive design attracts a diverse audience, making it accessible on various devices. Additionally, the dedicated section for offers and discounts appeals to budget-conscious customers, encouraging more purchases.

3. Technical Discussion

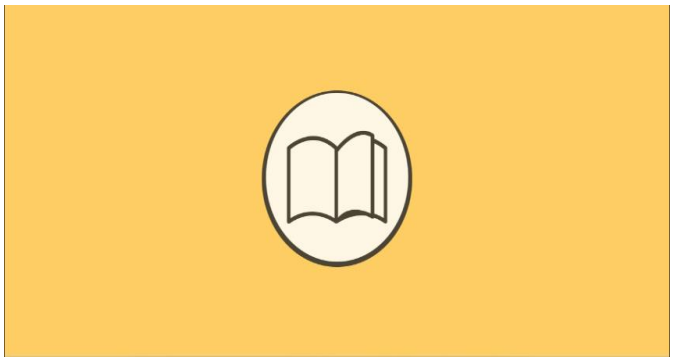
3.1) Student 1

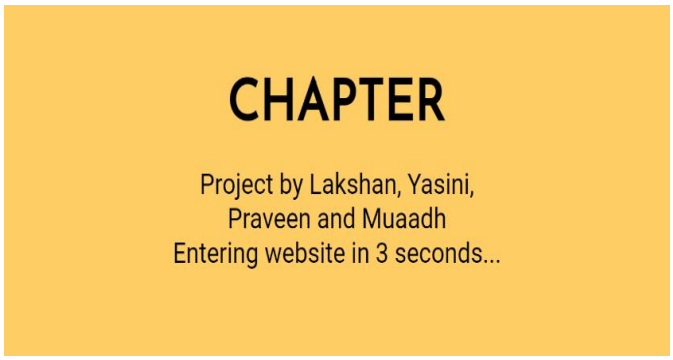
Presentation Page

course work > javascript > JS Presentation.js > ...

```
// Creates a countdown on a webpage from 5 to 0
let countdown = 5;
let countdownElement = document.getElementById('countdown');

//updating the countdown every second
let countdownInterval = setInterval(() => {
  countdown--;
  countdownElement.textContent = countdown;
}, 1000);

//After 5 seconds redirects the user to the "Home.html" page
setTimeout(function() {
    window.location.href = "Home.html";
  }, 5000);
```

Webpage's countdown starts with 5 seconds. The countdown that shows on the webpage is created by updating the text once every second using the'setInterval()' function. The user automatically navigates to the "Home.html" page after 5 seconds using the "setTimeout()" function. Displays the website logo first, followed by the title "CHAPTER" and the names of the team members are displayed.





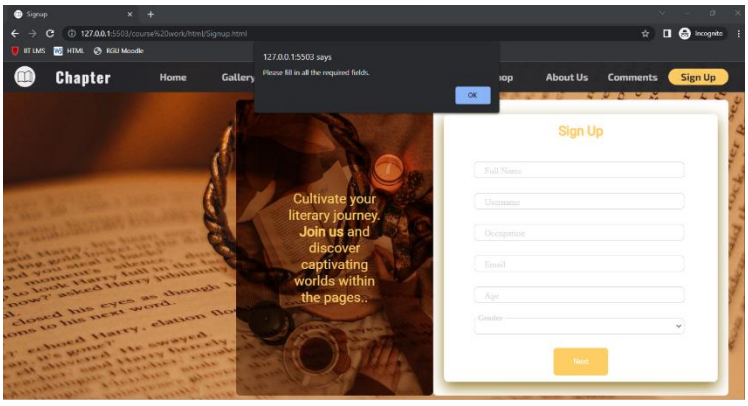
Signup Page

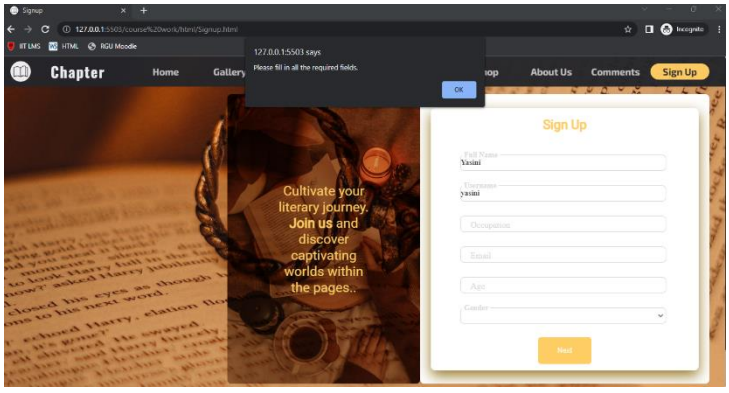
1. Form Validation and Submission:

Step 1 Form Validation:

```
// Get references to the forms and popup
var step1Form = document.getElementById('step1');
var step2Form = document.getElementById('step2');
var popup = document.querySelector('.popup');
var inner = document.getElementById('inner')

// Add event listener to Step 1 form submit
step1Form.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent form submission

// Validate Step 1 form fields
  var name = step1Form.elements.name.value.trim();
  var user = step1Form.elements.user.value.trim();
  var work = step1Form.elements.work.value.trim();
  var mail = step1Form.elements.mail.value.trim();
  var age = step1Form.elements.age.value.trim();
  var gender = step1Form.elements.gender.value;

  if (name === '' || user === '' || work === '' || mail === '' || age === '' || gender === '') {
    alert('Please fill in all the required fields.');
    return;
  }
```

When a form is submitted, event.preventDefault() is used to stop it from submitting immediately. The function then retrieves the data entered by the user in each field (name, user, work, mail, age, and gender) and stores them in the appropriate variables. The user will be asked from an alert to complete all needed fields before continuing if any of these fields are left empty.

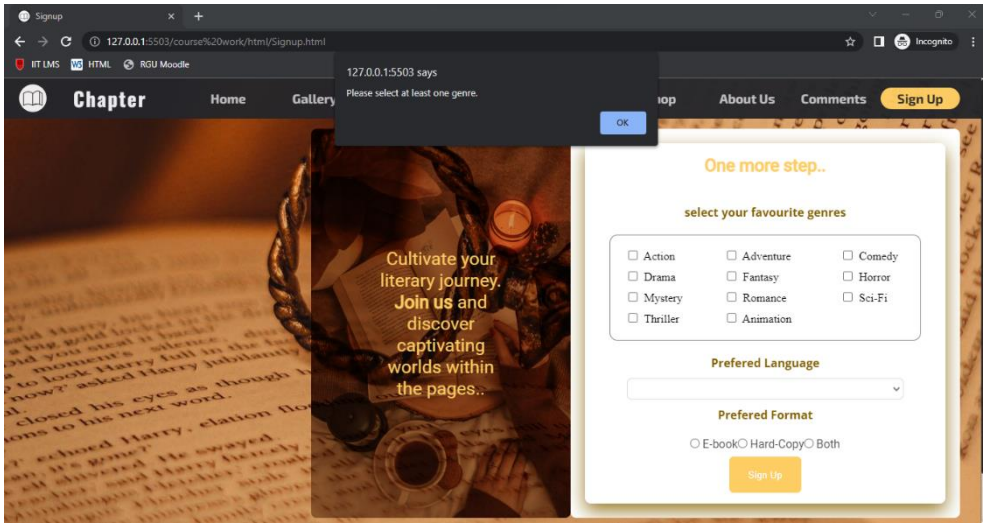


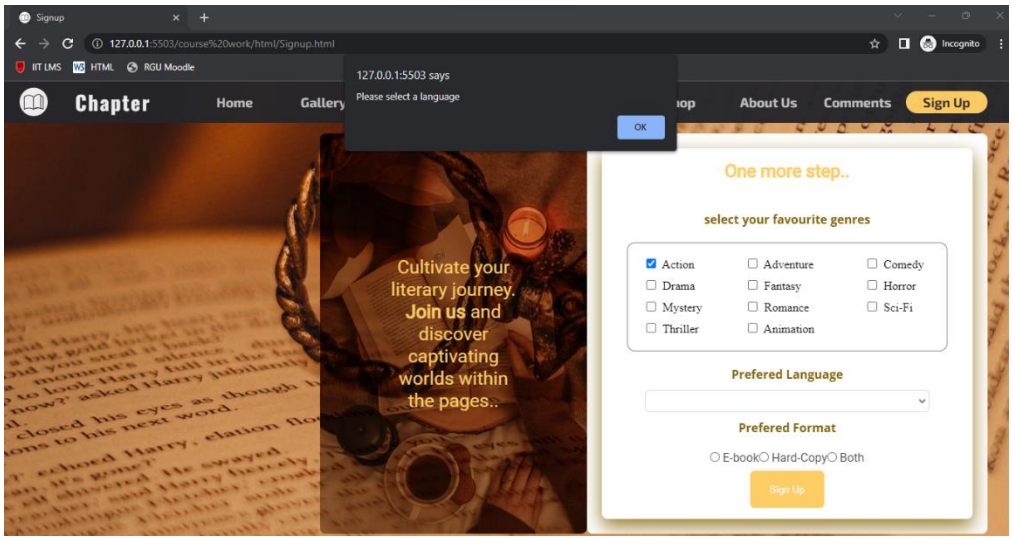


Step 2 Form Validation:

```
// Add event listener to Step 2 form submit
step2Form.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent form submission
// Get selected genres
var genres = [];
var checkboxes = step2Form.elements.genre;
const languageSelect = document.getElementById('Language');
const radioButtons = document.getElementsByName("option");
    let isChecked = false;
for (var i = 0; i < checkboxes.length; i++) {
  if (checkboxes[i].checked) {
    genres.push(checkboxes[i].value);
  }
}
if (genres.length === 0) {
  alert('Please select at least one genre.');
  return;
}
if (languageSelect.value === '') {
  alert('Please select a language');
  return;
}
for (const radioButton of radioButtons) {
  if (radioButton.checked) {
      isChecked = true;
      break;
  }
}
  if (!isChecked) {
      alert("Please select one option (E-book, Hard-Copy, or Both).");
      return false; // Prevent form submission
  }
```

The Step 2 form includes checkboxes for wanted genre selection, a dropdown for preferred language selection, and radio buttons for format selection. Then checks if at least one genre and a language has been chosen when the user submits the form. An alert is displayed to let the user know if any of these requirements is not satisfied.





2. Dynamic Step Form:

The dynamic step form allows the user to move from Step 1 to Step 2 without having to reload the website. It provides a smooth user experience by keeping the user engaged within the same page.

```
// Hide Step 1 form
  step1Form.style.display = 'none';

// Show Step 2 form
  step2Form.style.display = 'flex';
});
```

When the user submits the Step 1 form and it's valid, in this code hide the Step 1 form and show the Step 2 form. To make the transition smooth, also hide the container that holds both forms.

3. Popup Message:

The popup message serves as a personalized confirmation to the user that their CHAPTER account has been successfully created. It appears after successful validation and submission of the Step 2 form.

Popup Styling and Animation:

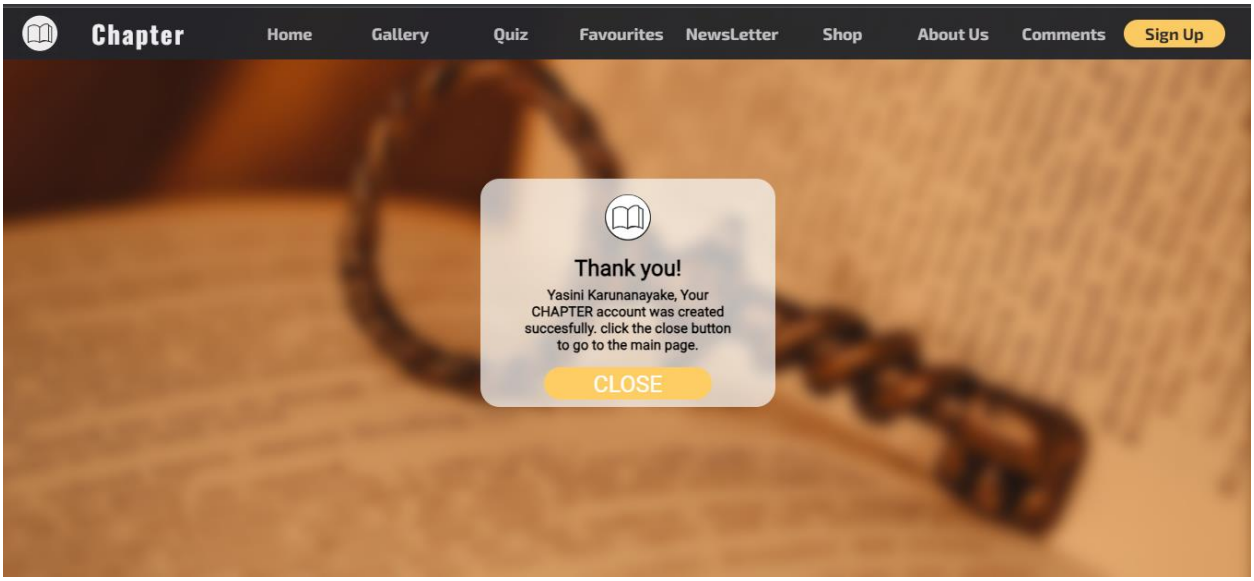
```css
/* Styling for popup box */
.popup .box {
  background-color: rgba(249, 252, 252, 0.7);
  border-radius: 21px;
  display: flex;
  flex-direction: column;
  position: absolute;
  width: 320px;
  height: 40%;
  justify-content: space-around;
  align-items: center;
  padding: 8px;
  animation: showPopupAnimation 0.3s ease;
}

/* Popup show animation */
@keyframes showPopupAnimation {
  0% { opacity: 0; transform: scale(0.7); }
  100% { opacity: 1; transform: scale(1); }
}

/* Popup hide animation */
@keyframes hidePopupAnimation {
  0% { opacity: 1; transform: scale(1); }
  100% { opacity: 0; transform: scale(0.7); }
}
```

The popup is normally hidden and only shows following a successful submission of the Step 2 form. It has a slightly transparent background so that you can see the content behind it. It contains a close button so users can dismiss it easily.

Popup Content:

```javascript
// Hide Step 2 form
  step2Form.style.display = 'none';
  inner.style.display = 'none';

// Show popup with personalized message
  var name = document.getElementById('fullname').value.trim();
  var userNameElement = document.getElementById('message');
  userNameElement.textContent = '' + name + ', Your CHAPTER account was created succesfully. click the close button to go to the main page. ';
  popup.style.display = 'flex';
});
```

The content of the popup is dynamically generated based on the user's provided Full Name (from Step 1 form). Extracts the value of the Full Name field and inserts it into the popup message: 'Your CHAPTER account was created successfully, ' + name + '. Click the close button to go to the main page.



Gallery Page

1. Book preview

Each thumbnail has an attribute data-name that stores a unique identifier for the book ("b-1" for Book 1). The popup is represented by an element with the class products-preview. There are separate elements for each book's details, each with an attribute data-target matching the data-name of the book.

Opening popup:

```
// Get the container for the product previews and all the preview elements inside it
let preveiwContainer = document.querySelector('.products-preview');
let previewBox = preveiwContainer.querySelectorAll('.preview');

// Add a click event listener to each element with the class 'gallery-item' inside the '.gallery' container
document.querySelectorAll('.gallery .gallery-item').forEach(products => {
  products.onclick = () => {
    // When a 'gallery-item' is clicked, display the 'products-preview' container
    preveiwContainer.style.display = 'flex';

    // Get the 'data-name' attribute of the clicked 'gallery-item'
    let name = products.getAttribute('data-name');

    // Go through each 'preview' element inside the 'products-preview' container
    previewBox.forEach(preview => {
      // Get the 'data-target' attribute of the current 'preview' element
      let target = preview.getAttribute('data-target');

    //Check if the 'data-name' attribute of the clicked 'gallery-item' matches the 'data-target' attribute of the current 'preview' element
      if (name == target) {
        preview.classList.add('active'); //If so,'preview' element corresponds to the clicked 'gallery-item'
      }
    });
  };
});
```

The name of the book is retrieved by JavaScript when a user clicks on a book thumbnail. Then, by adding the "active" class, it searches and displays the relevant book details using the data-target attribute inside the popup. This action also makes the products-preview modal visible by setting its display property to "flex".

Closing the Popup:

```
// Add a click event listener to each 'close-button' element inside a 'preview' element
previewBox.forEach(close => {
  close.querySelector('.close-button').onclick = () => {
    // When the 'close-button' is clicked, remove the class 'active' from its parent 'preview' element
    close.classList.remove('active');

    // Hide the 'products-preview' container by setting its display style to 'none'
    preveiwContainer.style.display = 'none';
  };
});
```

The popup can be closed by clicking the close button represented by the. close-button class. Removes the active class from the book details and sets the display property of the products-preview modal to none , hiding it from view.

3.2) Student 2

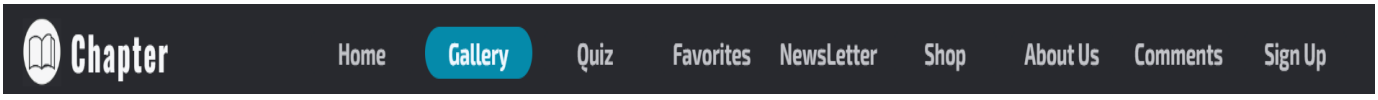
Main

Main page doesn't contain any functionalities. The information about the webpage is designed using html and CSS. A video is included.

Navigation

Navigation bar is implemented using functionalities of the current page active and CSS hover effects. The functionalities are explained in the particular title

This represents the current page.



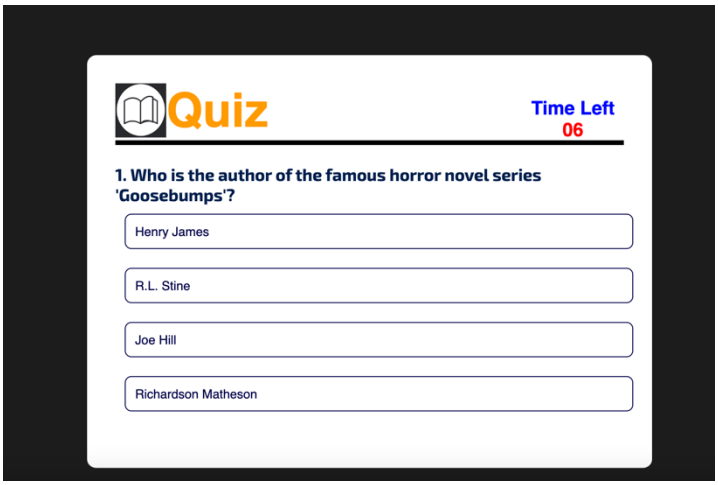
This represents the mouseover another page.

```
<li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
    " onmouseout="nav_back_highliter()" href="../html/Gallery.html"><button
        class="nav_button">Gallery</button></a></li>
```

This code snippet represents the mouseover and mouseout of the Gallery button effect. This is linked to the script file which will add classlist of the CSS file accordingly.

Quiz

The quiz part has functionalities to start, select answers for each question and displaying the final results. Once the answer is clicked, other answers are disabled and a function of countdown is set.



This is on progress quiz platform.

```
function showQuestion() {
    resetState();
    let currentQuestion= questions[currentQuestionIndex];
    let questionNo = currentQuestionIndex + 1;
    questionNumber.innerHTML = questionNo +". "+currentQuestion.question;

    currentQuestion.answers.forEach(answers=>{
        const button=document.createElement("button");
        button.innerHTML = answers.text;
        button.classList.add("answers");
        answerBtn.appendChild(button);
        if (answers.correct){
            button.dataset.correct = answers.correct;
        }
        button.addEventListener("click",selectAnswer);
    })
}
```

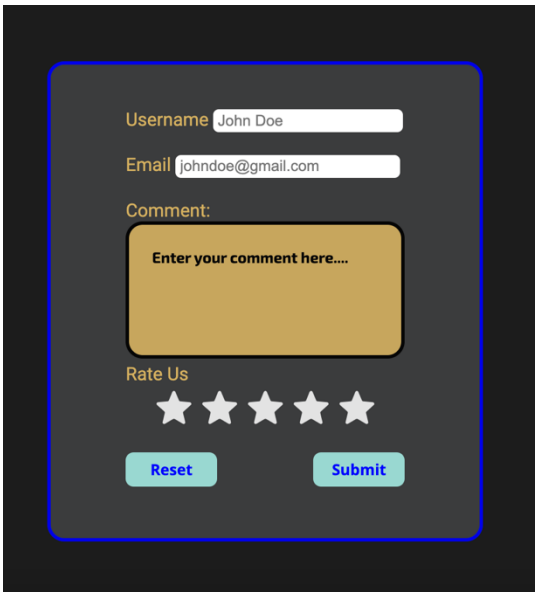
This will add the answers and the question for the platform for each next button click. resetState function is to reset the platform with new questions and answers.

```
function selectAnswer(e){
    const selectedBtn = e.target;
    const isCorrect = selectedBtn.dataset.correct== "true";
    clearInterval(counter);
    if(isCorrect){
        selectedBtn.classList.add("correct");
        score+=2;
    }else{
        selectedBtn.classList.add("incorrect");
        if (score>0){
        score--;
        }
    }
    Array.from(answerBtn.children).forEach(button =>{
        if(button.dataset.correct=="true"){
            button.classList.add("correct");
        }
        button.disabled=true;
    });
    nextBtn.style.display="block";
}
```

This code snippet is for display if the answer is correct or wrong. The score is added or deleted accordingly. Then the answers are displayed.

Comment

For the comments section, the name and email are validated using script functions as required. The empty comment and empty rating also validated using functions.




```
document.getElementById("submit").onclick=function(){

    validateEmail();
    validateTextInputs();
    if((rateComment=="")){
        alert("Select a rating option")
    }else{
    if(validatedEmail==true && inputs==true){
            alert("Dear "+usrName+", Thank you very much for your feedback. You rated our site as "+rateComment+".\n"+"Your comment was "+comment+".");
    }
    }
}
```

This code snippet shows that when the submit button is clicked, it will validate each field and text inputs. Then gives an alert of the details.

3.3) Student 3

The About Us page, book-related item buying page (Shopping-Cart), maximizing and minimizing the font size of the Shopping Cart page are mainly implemented as the task of student 3.

The Shopping cart consists cards which are created with the use of JSON and the Java Script Document Object Model (Dom) using a for each loop to ease the process of writing the html code to create cards.

```
// Loop through  items in the 'data' array using forEach and to provide the item and its index
as an arguments
data.forEach((dt, index) => {
  // For each item, append the following HTML code for the 'card_container' element
  card_container.innerHTML += `
  <div class="card">
   <img class="img_card" src="${dt.image}">
   <div class="detail_container">
       <h3 class="card_title"> ${dt.title}</h3>
       <h4 class="card_price">&pound; ${dt.price}</h4>
   </div>


   <button class="card_button" data-name="product1" onclick="handleBuy(${index}); index_finder
   = give_index(${index}); close_side_bar();">Buy Now</button>

</div>`;

})
```

The Item Add function is used to add items to the cart. This function checks whether the user-selected item already exists in the cart, and if it does not, it will add that item. when the user confirms add to cart then it shows the popup message with the details of the product on the shopping cart page.

```
// Function to add an item to the cart
function itemAdd() {
    // Checking if the item is already in the cart
   for (let a = 0; a < arr_item_index.length; a++) {
     if (index_finder == arr_item_index[a]) {
       window.alert('Item is already there in the cart !!!');
       return index_finder;
     }
   }
// To add the item to the cart and to update the cart's UI
   window.alert('Item has successfully addeed to cart !!!');
   cart_item_count += 1;
   arr_item_index.push(index_finder);
   total_quantity.innerHTML = String(cart_item_count);

   cart_array.push(
       {
        index:index_finder,
        image:data[index_finder].image,
        title:data[index_finder].title,
        quantity: 1,
        price : data[index_finder].price
       }
        )

}
```

When the user adds an item to the cart, it will add the item-related details to a cart array, and when the user clicks on the cart, it will be displayed using a for-each loop for the cart array and by writing the details using DOM. This method calls find_total_price method to calculate the total price of added items and display it when opening the side bar.

```
// This is function to display the side bar and its contents based on cart_array
const get_side_bar = (dk) => {
  side_bar.style.display = 'block';

  item_side_bar.innerHTML = '';


  cart_array.forEach((dk, index) => {
    // Purpose is to generate HTML elements for each item in the cart_array
    item_side_bar.innerHTML += `<div class='single_detail rm${index}'>
     <img src= ${dk.image} class='side_image'>
     <h3 class='side_book_title'>${dk.title}</h3>
     <div class="quantity">
     <form action="/action_page.php">
       <input type="number" id="quantity" name="quantity" min="1" max="999" value=${dk.quantity}
       class="quantity${index}" onchange="set_changed_val()">
     </form>
     </div>
     <button class="item_side_button" onclick="remove_from_cart(${index})";>Remove</button>
     </div>`;
  })

  // This is the function to calculate and display total price of items in the cart
  find_total_price();
}
```

The find_total_price method iterates through the cart_array and it gets the price of each item in the cart and that price is added to the total price. After calculating the total prices, it will be displayed in both side pane and checkout form using DOM.

```
// This is the function to calculate and to return the total price of all items in the cart
function find_total_price() {
  let total_price = 0;

  for (let j = 0; j < cart_array.length; j++) {
    if (cart_array[j]) {
      let item_price = (cart_array[j].quantity * parseFloat(cart_array[j].price)).toFixed(2);
      total_price = (parseFloat(total_price) + parseFloat(item_price)).toFixed(2);
    }
  }

   // Update the total price which displays in the UI
  let total_price_val = document.querySelector('.total_val');
  total_price_val.innerHTML = `Total: &pound;${total_price}`;

  let tot_bill_price = document.querySelector('.tot_bill_price');
  tot_bill_price.innerHTML = `Total Price: &pound;${total_price}`;

  return total_price;
}
```

The user will be able to change the quantity of the items added to the cart in the cart side pane. The set_changed_val function iterates through the cart_array, gets the current value of the quantity of each item in the sidebar cart, and updates the cart_array. In the subsequent process, it calls the find_total_price method to update the total price.

```
// This is the fuction to update the quantity of items in the cart_array and find the total
price
function set_changed_val() {
  for (let i = 0; i < cart_array.length; i++) {
    let quantity_item = document.querySelector(`.quantity${i}`);
    if (quantity_item) {
      let a = quantity_item.value;
      cart_array[i].quantity = a;
    }
  }

  find_total_price();
}
```

The remove_from_cart function is used to remove an item based on its index. After removing the item, the total price will be deducted, and the updated items in the cart will be printed using DOM. Following that, the total price of the sidebar cart and main pane is updated by using the find_total_price method.

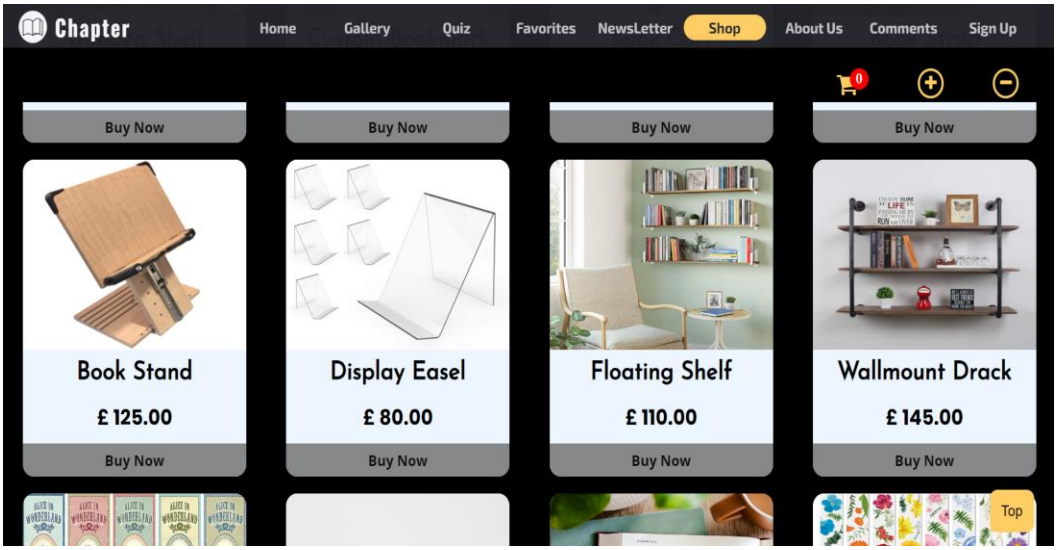
```
// This is function to remove an item from the cart based on its index
function remove_from_cart(index) {
  let remove_item = document.querySelector(`.rm${index}`);
  remove_item.remove();

  let price_reduce = (cart_array[index].quantity * parseFloat(cart_array[index].price)).toFixed(2);
  delete cart_array[index];
  delete arr_item_index[index];

  // This the fuction to re-calculate total price and update cart item count
  cart_array.forEach((dt, index_final) => {
    if (dt) {
      bill_data.innerHTML += `<div class='final_bill rm_final${index_final}'>
        <img src= ${dt.image} class='side_image final_bill_image'>
        <h3 class='unit_price'>&pound;${dt.price}</h3>
        <h3>${dt.quantity}</h3>
        <h3>&pound;${((dt.quantity * dt.price * 100) / 100).toFixed(2)}</h3>
      </div>`;
    }
  });

  cart_item_count -= 1;
  let total_quantity = document.querySelector('.total_quantity');
  total_quantity.innerHTML = `${cart_item_count}`;

  find_total_price();
}
```

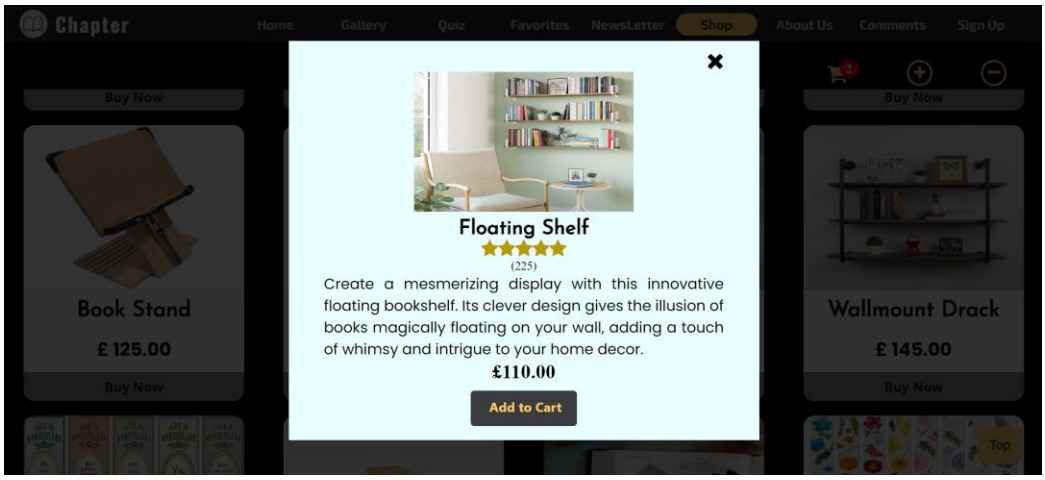
The submit form method is used to validate the checkout form when the user clicks on proceed payment. It produces a pop-up message that contains the details of the purchased items will be displayed. The regex

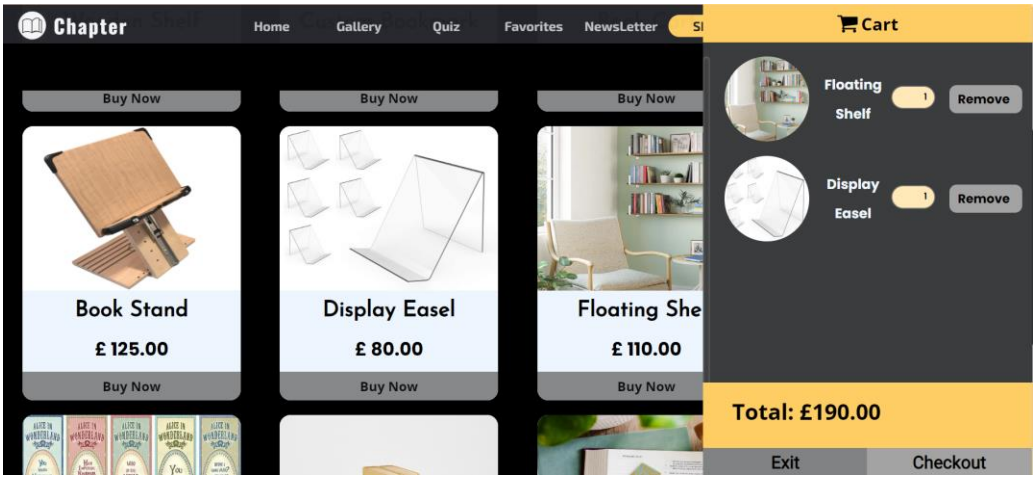
methods are used for validations.

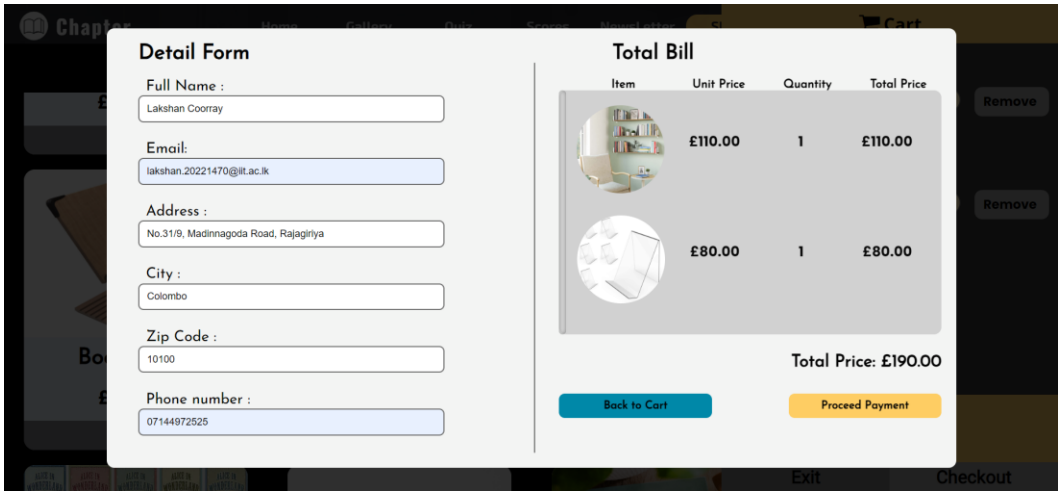
```
// This is the function to submit the form and perform validations
function SubmitForm() {

  let errorStat = false;

  formElements.forEach(element => {
    const { input, errorMsg, errorMsgText } = element;
    const value = input.value.trim();

    // This is the check if the input is empty, if yes, display the error message
    if (value === "") {
      errorMsg.innerHTML = errorMsgText;
      errorStat = true;
    } else {
      errorMsg.innerHTML = "";
    }
  });

  // For the validatation of the email address using a regular expression
  const emailInput = document.querySelector('#email');
  const emailErrorMsg = document.querySelector('#email_msg');
  const emailValue = emailInput.value.trim();

  const emailValidator = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!emailValidator.test(emailValue)) {
    emailErrorMsg.textContent = "Please enter a valid email address!!!";
    errorStat = true;
  }

  // Validating the item zipcode
  const zipInput = document.querySelector('#zip');
  const zipErrorMsg = document.querySelector('#zip_msg');
  const zip_value = zipInput.value.trim();

  const zipValidator = /^[\d+]\d+$/;
  if(!zipValidator.test(zip_value)){
    zipErrorMsg.textContent = "Please enter a valid zip code consisting only numbers";
    errorStat = true;
  }

    // Validating the phone number using a regular expression
  const phoneInput = document.querySelector('#phone_number');
  const phoneErrorMsg = document.querySelector('#phone_msg');
  const phoneValue = phoneInput.value.trim();

  const phoneValidator = /^[\d+]\d+$/;

  if (!phoneValidator.test(phoneValue) || phoneValue.length < 10) {
    phoneErrorMsg.textContent = "Please enter a valid phone number consisting of only numbers!!!";
    errorStat = true;
  }
```

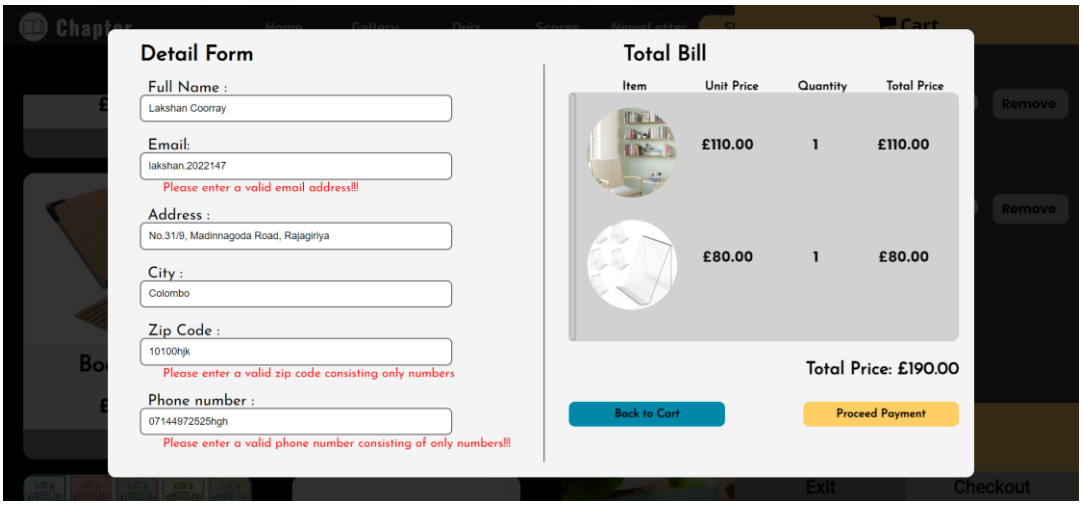
```
// Purpose if to display an alert if any errors are found in the form
if(errorStat){
  window.alert('Please fill the form corretly to proceed the payment !!!');
}else{
 // Hide the form and the sidebar and to display a confirmation message
  let deleteForm = document.querySelector('.checkout_form');
  deleteForm.style.display = 'none';

  let deleteSidebar = document.querySelector('.side_bar');
  deleteSidebar.style.display = 'none';

  let conformationMsg = document.querySelector('.conformatioMsg');
  conformationMsg.style.display = 'flex';

   // This is to create a message with the customer's name and ordered items
  const cusName = document.querySelector('#first_name');
  const requiredMassage = document.querySelector('.lastPop');
  requiredMassage.innerHTML = `Dear ${cusName.value}, you have ordered`;

  cart_array.forEach((dt, index)=>{
    if((cart_array.length - 1) != index){
      requiredMassage.innerHTML += ` ${dt.quantity} ${dt.title} at a cost of &pound;${dt.price} each and`;
    }else{
      requiredMassage.innerHTML += ` ${dt.quantity} ${dt.title} at a cost of &pound;${dt.price} .`;
    }

  })
```

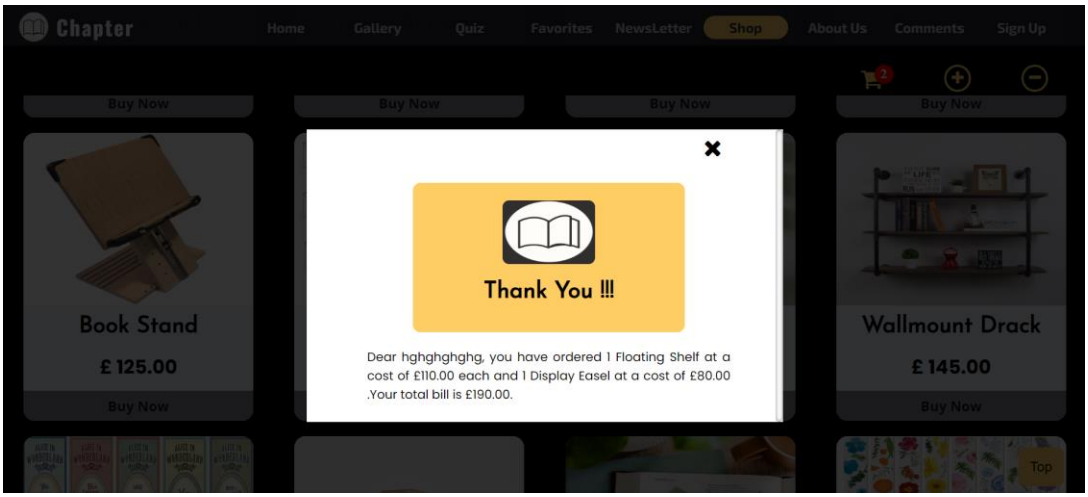



Chapter
Home
Gallery
Quiz
Favorites
NewsLetter
Shop
About Us
Comments
Sign Up
Buy Now
Book Stand
£ 125.00
Floating Shelf
(225)
Create a mesmerizing display with this innovative floating bookshelf. Its clever design gives the illusion of books magically floating on your wall, adding a touch of whimsy and intrigue to your home decor.
£110.00
Add to Cart
Wallmount Drack
£ 145.00
Top


Chapter
Home
Gallery
Quiz
Favorites
NewsLetter
Buy Now
Book Stand
£ 125.00
Display Easel
£ 80.00
Floating She
£ 110.00
Cart
Floating Shelf
1
Remove
Display Easel
1
Remove
Total: £190.00
Exit
Checkout


Detail Form
Full Name :
Lakshan Coorray
Email:
lakshan.20221470@iit.ac.lk
Address :
No.31/9, Madinnagoda Road, Rajagiriya
City :
Colombo
Zip Code :
10100
Phone number :
07144972525
Total Bill
Item
Unit Price
Quantity
Total Price
£110.00
1
£110.00
£80.00
1
£80.00
Total Price: £190.00
Back to Cart
Proceed Payment





The increaseFontSize method is used to increase the font size of the text in the shopping cart web page when plus button is pressed. Maximum sizes are allocated for different text levels. For example, for item name in the cards the maximum font size is given as 33. First it checks whether the text size is less than 33. If it is less than that, then it increments one to the font size and allocate the font size using the style property.

```
// This is the function to increase the font size of specified elements
function increaseFontSize(className, className2, className3) {
  const cardTitleElements = document.querySelectorAll(className);
    // For inreasing font size of elements with class className
  cardTitleElements.forEach((cardTitleElement) => {
    const fontSizeString = window.getComputedStyle(cardTitleElement).fontSize;
    let fontSize = parseInt(fontSizeString);

    if (fontSize < 33) {
      fontSize++;
      cardTitleElement.style.fontSize = `${fontSize}px`;
    }
  });
// To increase font size of elements with class className2
  const cardPriceElements = document.querySelectorAll(className2);
  cardPriceElements.forEach((dt) => {
    const fontSizeString = window.getComputedStyle(dt).fontSize;
    let fontSize = parseInt(fontSizeString);

    if (fontSize < 33) {
      fontSize++;
      dt.style.fontSize = `${fontSize}px`;
    }
  });
```

```
  // To increase font size of elements with class className3
  const BuyNow = document.querySelectorAll(className3);
  BuyNow.forEach((dt) => {
    const fontSizeString = window.getComputedStyle(dt).fontSize;
    let fontSize = parseInt(fontSizeString);

    if (fontSize < 28) {
      fontSize++;
      dt.style.fontSize = `${fontSize}px`;
    }
  });

}
```

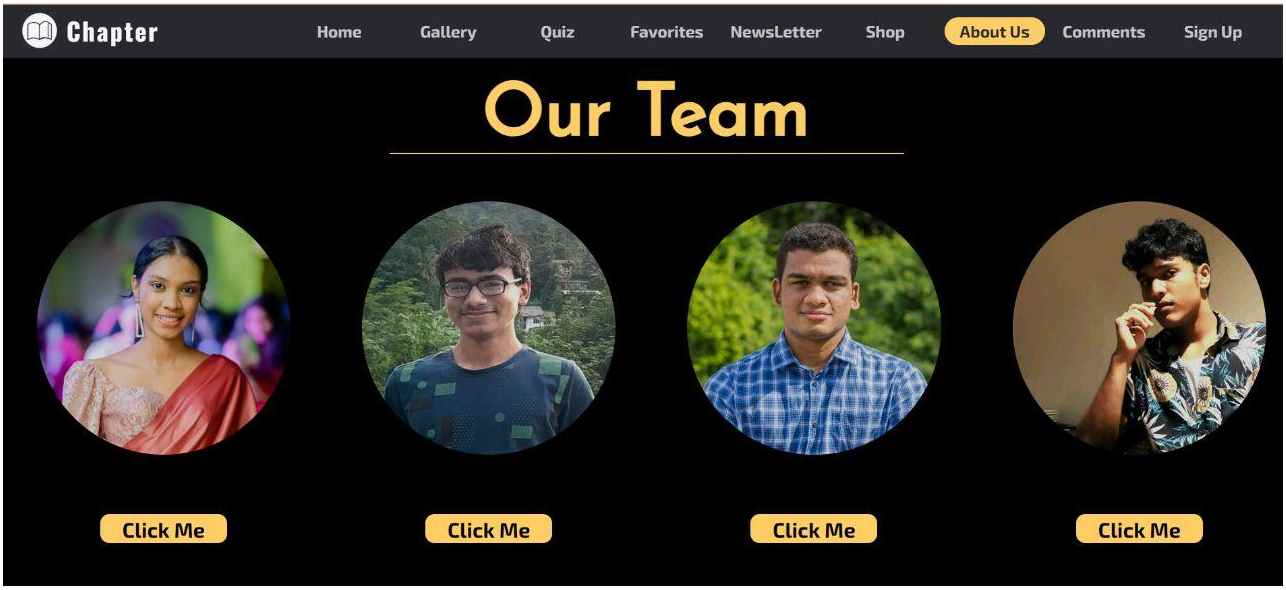
The decreaseFontSize method is used to decrease the font size of the text on the shopping cart web page when the minus button is pressed. Minimum sizes are allocated for each font size as above. Then it checks the current font size. If it is greater, it will decrement the font size by one, and using the font style properly, the new decremented font style will be updated.

```
// The function to decrease the font size of particluar elements
function decreaseFontSize(className, className2, className3) {
  const cardTitleElements = document.querySelectorAll(className);

  // To the decreasement of the font size of elements with class className
  cardTitleElements.forEach((cardTitleElement) => {
    const fontSizeString = window.getComputedStyle(cardTitleElement).fontSize;
    let fontSize = parseInt(fontSizeString);

    if (fontSize > 15) {
      fontSize--;
      cardTitleElement.style.fontSize = `${fontSize}px`;
    }
  });

  // To decrease font size of elements with class className2
  const cardPriceElements = document.querySelectorAll(className2);
  cardPriceElements.forEach((dt) => {
    const fontSizeString = window.getComputedStyle(dt).fontSize;
    let fontSize = parseInt(fontSizeString);

    if (fontSize > 15) {
      fontSize--;
      dt.style.fontSize = `${fontSize}px`;
    }
  });
```

```
    // Decrease font size of the elements with class className3
   const BuyNow = document.querySelectorAll(className3);
   BuyNow.forEach((dt) => {
     const fontSizeString = window.getComputedStyle(dt).fontSize;
     let fontSize = parseInt(fontSizeString);

     if (fontSize > 15) {
       fontSize--;
       dt.style.fontSize = `${fontSize}px`;
     }
   });

}
```

The about us page is created by including four images of the group members, and their names and task are displayed when the mouse moves over a picture. When the user moves over the picture, the style property of the selected division is updated as flex to display student details. When the user moves the cursor from the image, the style property of the selected division is updated to none to hide the student details.

```
// Function to show the details of a perticluar student by changing the CSS display to 'flex'
function show_detail(student_num){
    // To get the DOM element which represent the student's details using the student number
    let detail_student = document.querySelector(`#student_${student_num}`);
    // Change the display property to 'flex' to show the specific student details
    detail_student.style.display = 'flex';

}
//  Function to remove the details of a special student by changing the CSS display property to 'none'
function remove_detail(student_num){
    // To get the DOM element which represents the student's details by using the student number
    let detail_student = document.querySelector(`#student_${student_num}`);
    // To change the display property to 'none' and to hide the specific student's details
    detail_student.style.display = 'none';
}
```



3.4) Student 4

Among the tasks that assigned to Student 4, one was to create a page where the user can see their past activity. As instructed by the module lecturer, this task required a well-formed XML file where the user's past was recorded, and the XML file was to be created by myself.

With the knowledge I gained from the Web module as well as the internet, I was able to deduce the XML file down to the below structure.

```
1  <?xml version="1.0" encoding="UTF-8"?>
   Bind to grammar/schema...
2  <books>
3    <book>
4      <name>Dune</name>
5      <author>Frank Herbert</author>
6      <image>../images/dune.webp</image>
7    </book>
8    <book>
```

Figure 0.1 XML Structure

The XML file consists of a super element called <books> and within, there are sub elements for each item named <book>. Within the <book> element lies the attributes of a book, its name, its author, and a path to the image file of the book cover.

The html that accepts this XML data was constructed so that only a minimum number of lines are needed, and the JavaScript code creates its own html elements to proceed further with the XML data.

```html
<main>

    <div id="container">
        <div id="card-container">

            <div id="divpref">
                <table id="table1" style="border-collapse:collapse" border="2" cellpadding="5">
                </table>
            </div>
        </div>
```

Figure 0.4 HTML block for the XML

```javascript
function loadXMLDoc() {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      prefDetails(this);
    }
  };

  xmlhttp.open("GET", "http://127.0.0.1:5500/xml/Favorites.xml", true);
  xmlhttp.send();
}
```

Figure 0.3 JavaScript for XML 1

```javascript
  function prefDetails(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var cardsContainer = document.getElementById("card-container");
  cardsContainer.innerHTML = "";
  var x = xmlDoc.getElementsByTagName("book");

  for (i = 0; i < x.length; i++) {
    var name = x[i].getElementsByTagName("name")[0]?.textContent;
    var author = x[i].getElementsByTagName("author")[0]?.textContent;
    var image = x[i].getElementsByTagName("image")[0]?.textContent;

    var card = document.createElement("div");
    card.className = "card";

    var img = document.createElement("img");
    img.src = image;
    img.alt = name;

    var cardContent = document.createElement("div");
    cardContent.className = "card-content";

    var bookName = document.createElement("h3");
    bookName.textContent = name;

    var bookauthor = document.createElement("p");
    bookauthor.textContent = author;

    cardContent.appendChild(bookName);
    cardContent.appendChild(bookauthor);

    card.appendChild(img);
    card.appendChild(cardContent);

    cardsContainer.appendChild(card);
  }
  }
```

Figure 0.2 JavaScript for XML 2

The XML is loaded into cards inside the html body using the JavaScript loadXMLDoc() function and the prefDetails() function accepts the XML data and using the JavaScript, I am able to manipulate the html container so that the XML data is loaded into the html and is displayed as cards as shown below.

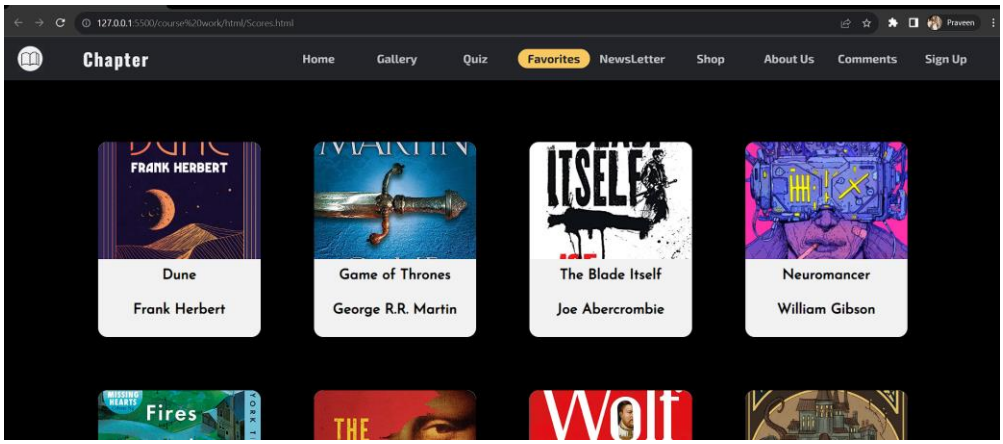


Figure 0.5 Favorites page

The other task was to create a newsletter in a different page. A newsletter is created using html forms and the user inputs are required to be validated.

```
let x = false;
let y = false;


function ValidateEmail(email) {
  if (/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/.test(email)) {
    return(x = true)

  }else {
    alert("You have entered an invalid email address!");
    return (false)
  }
}


function ValidateName(name) {

  if (/^[A-Za-z\s]*$/.test(name)) {
    return(y = true)

  }else {
    alert("You have entered an invalid name!")
    return (false)

  }
}


function TrueDetails() {
  if (x == true && y == true) {
    alert("Dear Anne, you have successfully subscribed for a personalized newsletter")
    return (true)
  }
}

```

Figure 0.6 Newsletter JavaScript

Once the html form for the newsletter was created, I designed the above JavaScript so that once the user inputs namely the user's name and email address are inserted, the functions ValidateEmail() and ValidateName() tests the email and the name of the user respectively. For the validation of inputs, my research led me to Regex/Regular Expressions patterns.

By comparing the inputs to the Regex patterns, I was successful in Validating the user inputs in the newsletter.

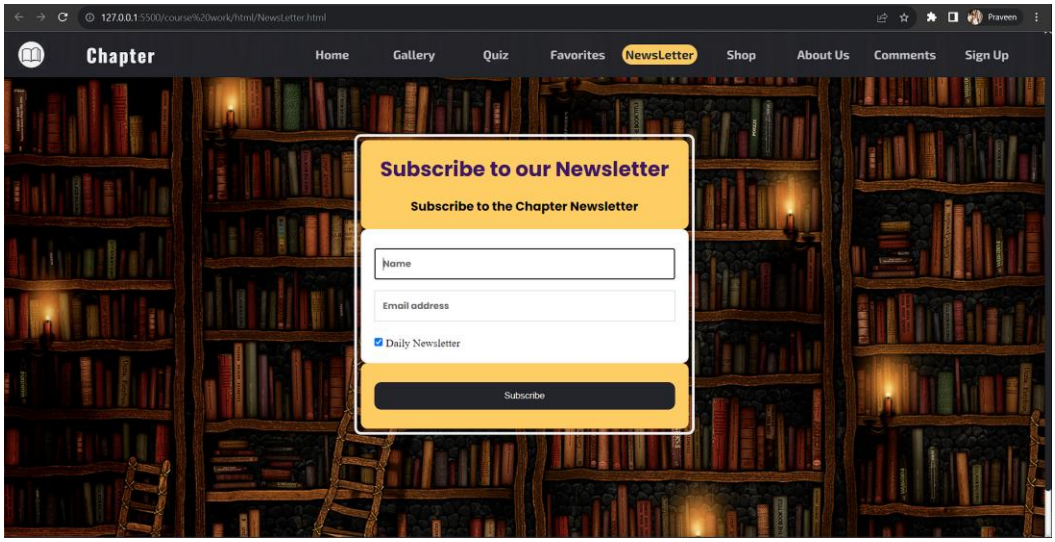
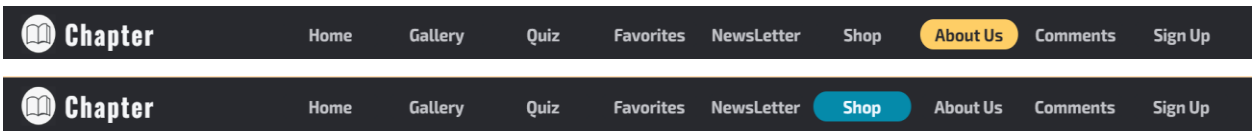


Figure 0.7 Newsletter Page

4. Discussion of UX/UI principles/Applications/Justifications

4.1) Navigation Techniques

The navigation bar was mainly developed by student 2 to facilitate efficient navigation between the pages of the website. The developed navigation allows the user to understand the current web page since the name of the current webpage is highlighted in the yellow color code that has been used throughout the web page to ensure the breadcrumb trail. The HTML, CSS and JavaScript code that has used to develop the navigation bar is given below.



HTML code

```
    <!-- Navigation menu -->
    <ul class="nav_ul">
        <li class="nav_list">
            <!-- Home link with a button -->
            <a class="nav_link" onmouseover="nav_highliter();
       " onmouseout="nav_back_highliter()" href="../html/Home.html"><button class="nav_button"
                id="button_1">Home</button></a></li>
<!-- Other navigation links which use buttons -->
        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
        " onmouseout="nav_back_highliter()" href="../html/Gallery.html"><button
                class="nav_button">Gallery</button></a></li>
                <!-- other navigation links ... -->
        <!-- ... -->

        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
        " onmouseout="nav_back_highliter()" href="../html/Quiz.html"><button
        class="nav_button">Quiz</button></a>
        </li>

        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
        " onmouseout="nav_back_highliter()" href="../html/Scores.html"><button
                class="nav_button">Scores</button></a></li>

        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
            " onmouseout="nav_back_highliter()" href="../html/NewsLetter.html"><button
            class="nav_button">
                NewsLetter</button></a></li>
```

```
        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
            " onmouseout="nav_back_highliter()" href="../html/ShoppingCart.html"><button
                class="nav_button ">Shop</button></a></li>

        <li class="nav_list"><a class="nav_link" href="#"><button class="nav_button current_page
        hello">About
                Us</button></a>
        <li>

        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
        " onmouseout="nav_back_highliter()" href="../html/Comments.html"><button
                class="nav_button">Comments</button></a></li>

        <li class="nav_list"><a class="nav_link" onmouseover="nav_highliter();
        " onmouseout="nav_back_highliter()" href="../html/Signup.html"><button
        class="nav_button">Sign
                Up</button></a></li>

    </ul>
```

Css code

```
header{
    display: flex;
    justify-content: space-between;
    align-items:center;
    height:9vh;
    background-color:var(--col_black);
    padding-left: 20px;
    padding-right: 30px;
    position:fixed;
    width: 100%;
    margin-bottom: 0;
    z-index: 1600;
    opacity:0.98;
}

.nav_ul{
    display: flex;
    list-style-type:none;
    transition: cubic-bezier(0.175, 0.885, 0.32, 1.275) 0.5s all;
}

.nav_list{
    margin-left:10px;


}
```

```css
.nav_button{
    height: 20px;
    background-color: var(--col_black);
    font-family: var(--exo2_font);
    color: #c2c3c8;
    font-size: 18px;
    height:30px;
    width:110px;
    text-align: center;
    border-style: none;
    border-radius: 15px;
    transition-duration: 0.5s;
    transition-property: all;
    transition-timing-function: ease-in-out;
    cursor: pointer;

}

.current_page{
    background-color: #FECD63;
    color:var(--col_black);
}

.current_page:hover{
    background-color: var(--col_blue);
    color: aliceblue;
}
```

```css
.current_page:active{
    background-color: #FECD63;
    color:var(--col_black);
    transition-duration: 0.4s;
    transition-timing-function: ease-in;
}

.nav_button:hover{
    background-color: var(--col_blue);
    color: aliceblue;
}

.nav_button:active{
    background-color: #FECD63;
    color:var(--col_black);
}

.header-topic{
    opacity:0.96;
}


.main_video{
  width: 100%;

}
```

```css
.main_div1{
    margin-top: -10px;
    height: 105vh;
    background-color: #24252a;
}

.chapter_h1{
    margin-left: -9%;
    color: #f6f7f9;
    font-family: var(--openSans_font);
    letter-spacing: 2px;
    font-size: 27px;

}
```

JavaScript code

```
function nav_highliter() {
    let current_button = document.querySelector(".current_page");
    if(current_button != null){
        current_button.classList.remove("current_page");
    }

  }

function nav_back_highliter() {
    let current_button = document.querySelector(".hello");
    current_button.classList.add("current_page");
}
let navButton = document.querySelector(".hamburgerButton");
let navBar = document.querySelector(".nav_ul");

let activeState = false;

function nav() {
  activeState = !activeState;
  navBar.classList.toggle("active", activeState);
  navButton.classList.toggle("active", activeState);
}

let navLinks = document.querySelectorAll(".nav_ul li a");
navLinks.forEach((link) => {
  link.addEventListener("click", () => {
    nav();
  });
});
```

4.2) Color balance/selection/consistency

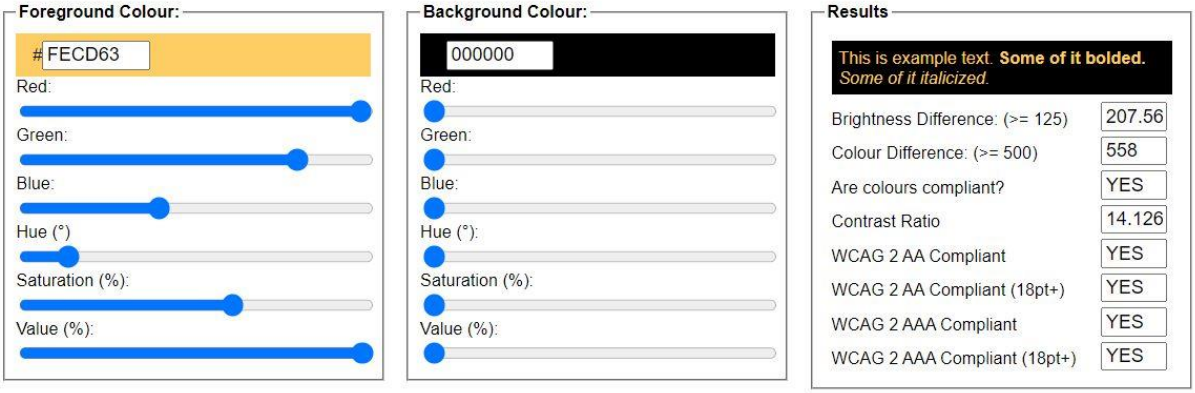
The Majority of the selected colors have met the color standards. Black, yellow and white based colors are mainly used. Black and yellow theme adds a lively and engaging atmosphere and the combination of black and white based colors improves the readability and the simplicity of the website. However, since this website is used to develop a book recommendation system the logo has designed by including an image of a book to highlight it is a book recommendation system and to ensure the consistency.The below CSS shows the code pattern used for color balance of the website.

```css
@import url('https://fonts.googleapis.com/css2?family=Caveat:wght@700&family=Exo&family=Exo+2:wght@700;900&family=Fira+Sans+Condensed:wght@700&family=Josefin+Sans&family=Open+Sans:wght@700&family=Oswald:wght@700&family=Poppins:wght@400;500;600;700&family=Roboto&family=Varela+Round&display=swap');

/* CSS Variables */
:root {
    --font1: 'Open Sans', sans-serif;
    --font2: 'Roboto';
    --colyellow: #FECD63;
    --colblue: #0088a8;
    --col_black1: #3b3c3d;
    --font3:'Exo 2', sans-serif;
    --poppins:'Poppins', sans-serif;
    --jossofin:'Josefin Sans', sans-serif;
    --fontWhite: white;
    --colMainBlack: black;
}
```

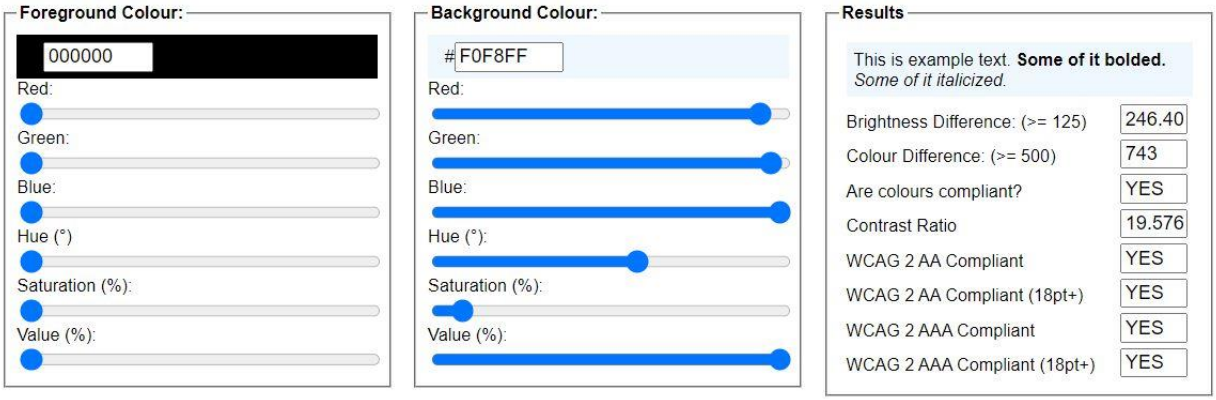
4.3) Color Contrast Test

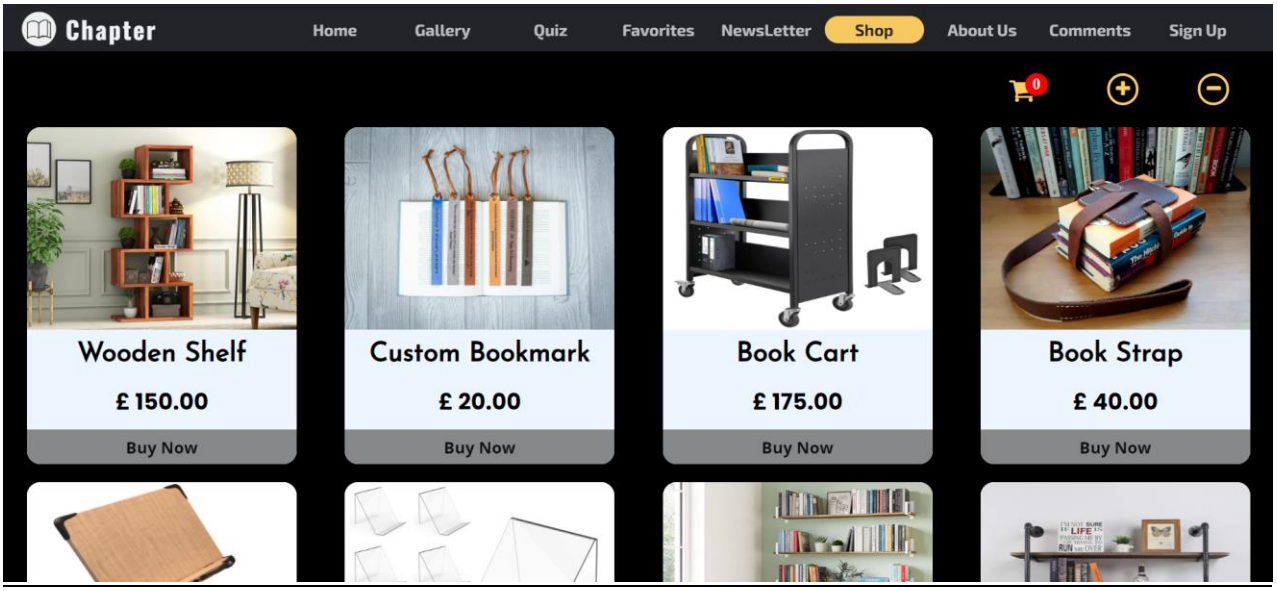
The color contrast test done has ensured that perfect compliant colors are used to design this web application. The below images show the results of the color contrast test.

All the main web pages

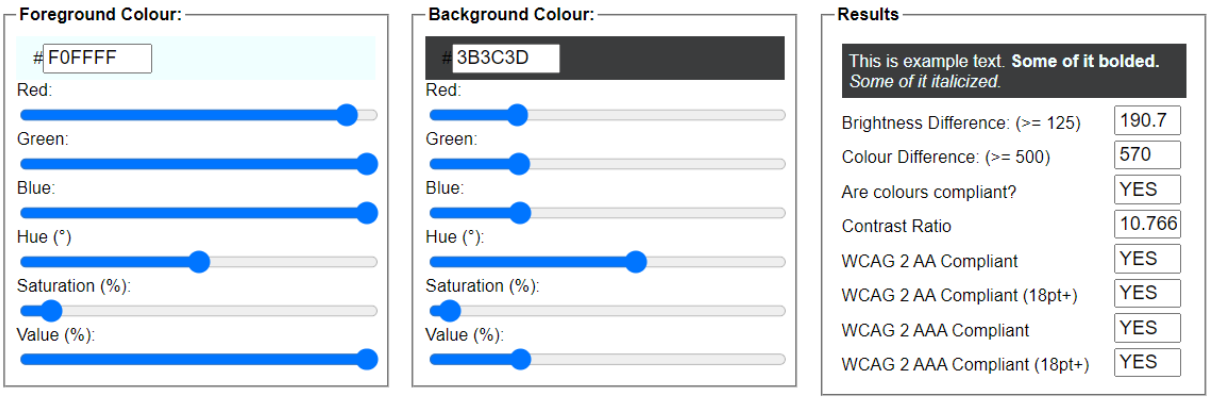


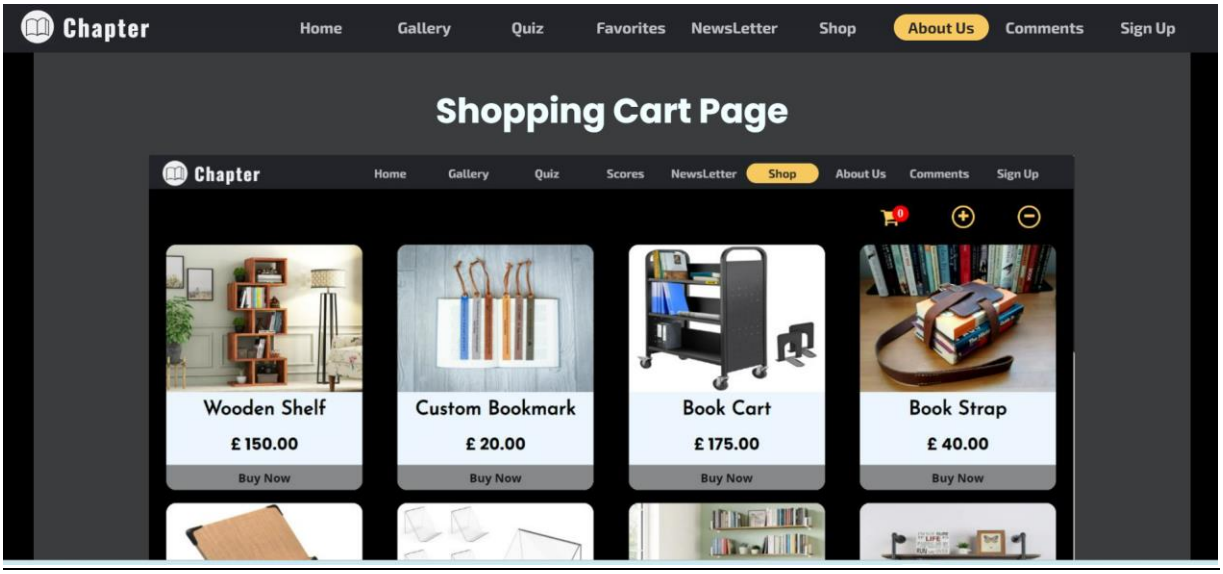
Shopping Cart Cards





Editor's Page





4.4) Typography / consistency

The interesting yet a difficult part of this coursework is that throughout every student's portion of the web page, there must always be a consistency in the theme, the fonts as well as the colors and that was the first topic to be discussed within our group. Few of the discussed examples are below;

The home page is the starting page of this web. The navigation bar and the body of the web page are of the same color theme and the background color is a custom color we chose. (Hex - #24252a).

Figure 0.1 Home Page

- Navigation Bar

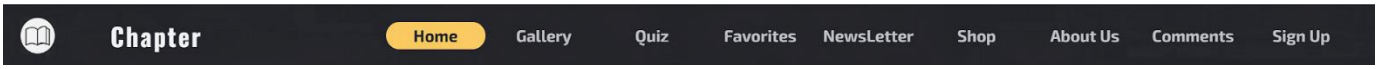


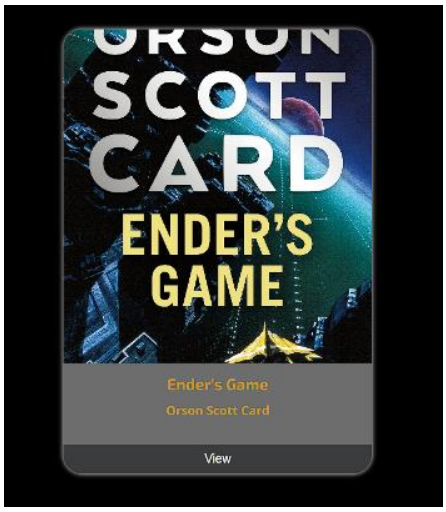
Figure 0.2 Navigation Bar

Throughout the navigation bar, the same font (Sans-serif) with 18 px with 9.9 px estimated x-height is applied to maintain the theme. When a specific navigation tab is clicked, the yellow marker is moved over to that tab. The consistent font was attained by the below CSS code.

```
.nav_button{
    height: 20px;
    background-color: var(--col_black);
    font-family: var(--exo2_font);  /*Font - sans-seriff *
    color: #c2c3c8;
    font-size: 18px;
    height:30px;
    width:110px;
    text-align: center;
    border-style: none;
    border-radius: 15px;
    transition-duration: 0.5s;
    transition-property: all;
    transition-timing-function: ease-in-out;
    cursor: pointer;

}
```

Figure 0.3 Navigation bar CSS

- Gallery



The choices of books within the gallery page are displayed in cards to make it more appealing to the viewer. Each card has an image covering approximately 75% of the card from the top to bottom and the bottom portion is reserved for the name of the book, and the author and all the fonts within the cards are Sans-serif with the book title being 18 px with 9.9 px estimated x-height and the author being 14px with estimated 7.7px x-height with the below CSS.

```css
/* Styling for the book title */
.book-title {
    margin: 0;
    font-size: 18px;
    font-family: var(--font3);
}

/* Styling for the book author and genre */
.book-author,
.book-genre {
    margin: 5px 0;
    font-size: 14px;
    font-family: var(--font1);
}
```

4.5) Accessibility

4.5.1) Text Accessibility

The language is specified in all the webpages. The titles are used which will allow users to identify each page uniquely. The headings are used in the hierarchy structure. Meta name is also specified. Consistency of the webpages are maintained throughout the website.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Us</title>
```

```
<div class="text-block">
    <h2 class="book-name">Dune</h2>
    <h3 class="Genre"><span>Genre:</span> Science Fiction</h3>
    <h3 class="Author"><span>Author:</span> Frank Herbert</h3>
    <h3 class="year"><span>Year:</span> 1965</h3>
    <h4 class="Description">"Dune" is a science fiction novel set
</div>
```

4.5.2) Image Accessibility

The alternate text for the images are used in entire website.

```
<img src="../images/linkedin.png" alt="linkedin" height="20%" width="20%"
<img src="../images/instagram.png" alt="instagram" height="20%" width="20%"
<img class="header-topic" src="../images/ChapterLogoFix.jpg" height="40vh" width="40vw" alt="Chapter Logo">
```

4.5.3) Accessibility Table

We didn't use any tables in the website. The accessibility is not set.

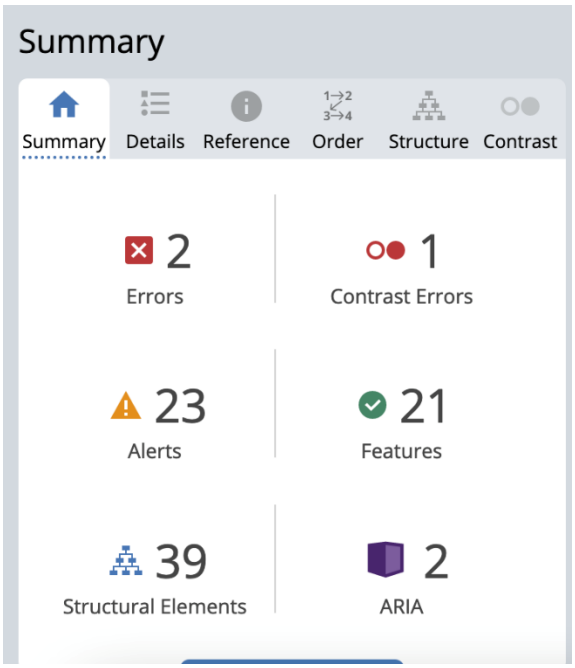
4.5.3) Accessibility Forms

The explicit labels for the input type are included the forms in all webpages. The field set and legend are used in some webpages. Required fields are not set as they must be validated using java script.

```
<label for="userName">Username</label>
<input type="text" class="inputs" id="userName" placeholder="John Doe"><br><br>
<label for="email">Email</label>
<input type="text" class="inputs" id="email" placeholder="johndoe@gmail.com"><br><br>
<label for="comment">Comment:</label>
<textarea name="comment" id="comment">Enter your comment here....</textarea><br>
```

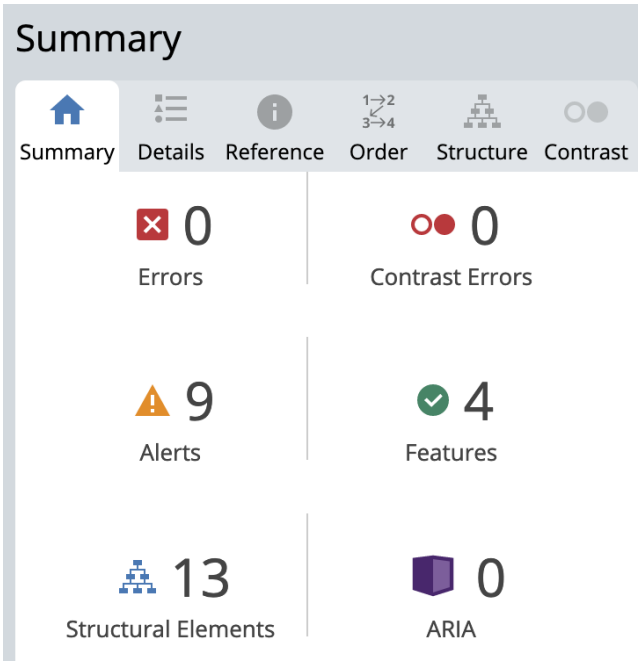
4.6) Test Report

1) Shopping Cart



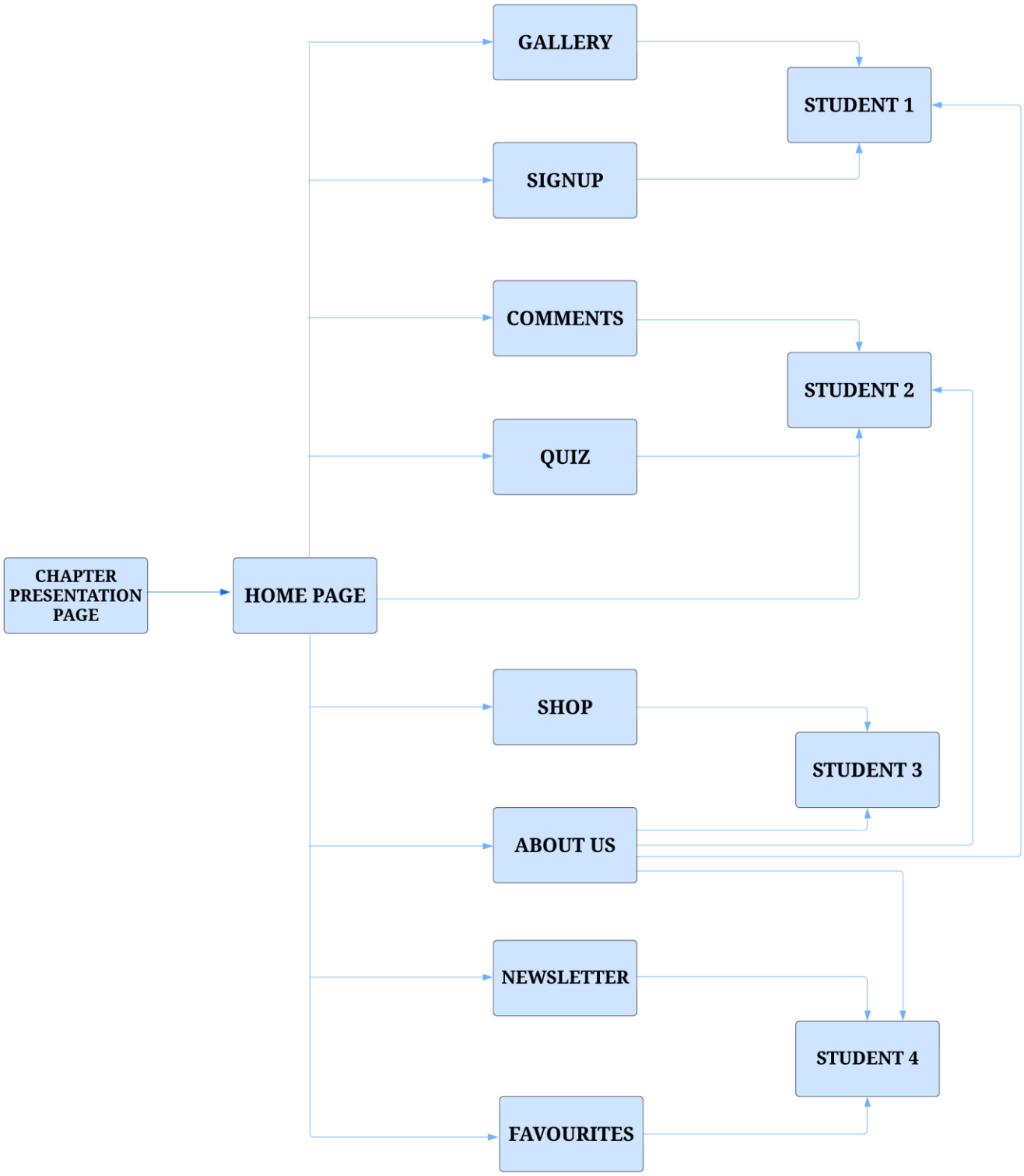
The above picture is the summary of the shopping cart page. The two errors shown are for an input type which doesn't have a label and for a heading which doesn't have any content within it. Those two are used in scripting file. There are some alerts which includes in the navigation bar. But it can be ignored as the navigation bar is functioning properly. The constraints of the structure of a website is followed throughout the webpage.

2) Quiz Page



This is the summary of the accessibility testing of the quiz page. There are no any errors. The alerts are shown for the navigation bar as mentioned. The constraints of the structure of a website is followed throughout the webpage.

4.7) Site Diagram



5. Self-Reflection

During this semester we had a group project of 4 members for the web developing module. We worked collaboratively to bring success to this project which is aimed to get the knowledge on client-side web developing.

5.1) Work carried out

The task was divided among the group members according to the specification provided as Student 1, Student 2, Student 3 and Student 4. The collective of these tasks bring a well-developed client-side website.

5.2) Collaboration

We had to communicate with each other time to time. So, we had meetings, brainstorming ideas, and other many methods which could improve our collaboration. As the task was divided, we had focused scopes. So, we didn't face many challenges until combining the website.

5.3) Challenges encountered

As we progressed initially it was fine. But as the project progress moves we encountered many challenges. We had a tight schedule to get to a meeting. Each student faced challenges according to their tasks. Student 1 had challenge of creating the thumbnail gallery page. Student 2 had challenge of creating navigation links. Student 3 had faced a major problem of creating the shopping cart with all accessibility. Student 4 had encountered challenge with creating favorite List. Creating the suitable functions for them was a harder part out of other functions. We had spent more time than expected as the challenges approached.

5.4) Lessons Learnt and Suggestions

Throughout the project, we gained many valuable technical skills. We learnt to work according to scheduled time and work as collaborative team. When each had problems, they get the feedback from others. In that way we improved our qualities of our website. We should have more meeting time with the members. This would have increased our time management. Moreover, with other chat messages we would have more communications. Ensuring all the team members are familiar would have prevented the absence of the particular member.

5.5) Conclusion

Working in a group project gives us the environment related to the industrial experience. Throughout the project we learnt to face challenges, find a solution for them, Giving suggestions and many other development skills. We are committed to applying the lessons learned in the future collaborations and look forward to further refining our skills in other fields also.

6. References

CodeHim. (2022). 15+ JavaScript Shopping Cart Examples with Demo. [online] Available at: https://www.codehim.com/collections/javascript-shopping-cart-examples-with-demo/.

www.tutorjoes.in. (n.d.). Shopping Cart - JavaScript. [online] Available at: https://www.tutorjoes.in/JS_tutorial/food_shopping_cart_in_js [Accessed 27 Jul. 2023].

www.youtube.com. (n.d.). Build a Shopping Cart with JavaScript – Project Tutorial. [online] Available at: https://youtu.be/cT_ZYrS3tKc [Accessed 6 Apr. 2023].

www.youtube.com. (n.d.). Ecommerce Website | Add to Cart | Delete from Cart | HTML, CSS & JavaScript. [online] Available at: https://youtu.be/sf_ac-dYh3w [Accessed 5 Apr. 2023].

www.youtube.com. (n.d.). Create a Ecommerce Website Using HTML CSS And JavaScript - JavaScript Working Shopping Cart. [online] Available at: https://youtu.be/18Jvyp60Vbg.

freeCodeCamp.org. (2021). How to Build a Sign Up Form with Floating Labels and Transitions Using Plain HTML and CSS. [online] Available at: https://www.freecodecamp.org/news/how-to-build-sign-up-form-with-html-and-css/ [Accessed 17 Jun. 2023].

www.youtube.com. (n.d.). Responsive Image Gallery with Search Box in HTML CSS & JavaScript. [online] Available at: https://youtu.be/jB6RmVoYotc [Accessed 20 Jun. 2023].

www.youtube.com. (n.d.). Image popup in html and css | awesome popup effect in js | Coding Tutorial. [online] Available at: https://youtu.be/kgLD27jrWRo [Accessed 20 Jun. 2023].

www.youtube.com. (n.d.). How To Make Form (Multi-Step) Using HTML CSS & JS | Create Form With HTML & CSS. [online] Available at: https://youtu.be/T76bbMVMX6M [Accessed 18 Jun. 2023]