وظيفة مادة برمجة ويب 2 BWP501

S23

تقدمة :

|  |  |  |
| --- | --- | --- |
| الاسم | الرقم الجامعي | الصف |
| يزن جوريه | Yazan\_174681 | C3 |

رابط الموقع :

https://yazan-174681.000webhostapp.com

1. Index.html

تحوي على الصفحة الرئيسية للموقع و تتألف من العناصر التالية

* + <div class="container"> يعرف عنصرًا تحتوي على جميع عناصر الصفحة.
  + <div class="top"> يعرف عنصرًا للجزء العلوي من الصفحة يحتوي على العنوان والقائمة العلوية.
  + <div class="title"> يعرف عنصرًا لعنوان المتجر.
  + <ul class="top-list"> يعرف عنصرًا للقائمة العلوية التي تحتوي على عناصر المساعدة والدعم.
  + <div class="mid"> يعرف عنصرًا للجزء الأوسط من الصفحة يحتوي على القائمة الجانبية وقائمة الكتب.
  + <div class="side"> يعرف عنصرًا للقائمة الجانبية.
  + <div class="f-1"> يعرف عنصرًا لزر "قائمة الرغبات".
  + <div class="f-2"> يعرف عنصرًا للقوائم الفرعية في القائمة الجانبية.
  + <div class="main"> يعرف عنصرًا للمحتوى الرئيسي للصفحة.
  + <div class="head"> يعرف عنصرًا لعنوان قسم الكتب المتاحة.
  + <div class="nav"> يعرف عنصرًا لشريط التنقل الذي يحتوي على قائمة التصفية وحقل البحث.
  + <ul class="filters"> يعرف عنصرًا لقائمة التصفية.
  + <div action="" class="search"> يعرف عنصرًا لحقل البحث.
  + <div id="booklist" class="book-list"> يعرف عنصرًا لقائمة الكتب المتاحة.
  + <div class="foot"> يعرف عنصرًا للجزء السفلي من الصفحة.
  + <div class="sett-b"> يعرف عنصرًا لزر الإعدادات.
  + <div class="ask-b"> يعرف عنصرًا لزر المساعدة.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>My Book Store</title>

    <link rel="stylesheet" href="./Css/Normalize.css">

    <link rel="stylesheet" href="./Css/style.css">

    <link rel="stylesheet" href="./Css/all.min.css">

</head>

<body>

    <div class="container">

        <div class="top">

            <div class="title">

                <i></i>

                <span>My Book store</span>

            </div>

            <ul class="top-list">

                <li>Help Center</li>

                <li>Our Support</li>

                <li id="r-status"></li>

            </ul>

        </div>

        <div class="mid">

            <div class="side">

                <div class="f-1">

                    <button id="wishlist" class="add-book-button"><i></i>My Wishlist</button>

                </div>

                <div class="f-2">

                    <ul >

                        <li><i class="fa-book"></i>Now Reading</li>

                        <li><i class="fa-globe"></i>Browse</li>

                        <li><i></i>Buy Books</li>

                        <li><i></i>Favourate Books</li>

                        <li><i></i>Wishlist</li>

                        <li><i></i>History</li>

                    </ul>

                </div>

                <div class="f-2">

                    <ul >

                        <li><i></i>Must Read Titles</li>

                        <li><i></i>Best Of List</li>

                        <li><i></i>Classic Novels</li>

                        <li><i></i>Non Action</li>

                    </ul>

                </div>

            </div>

            <div class="main">

                <div class="head"><span>Browse Available Books</span></div>

                <div class="nav">

                    <ul class="filters">

                        <li>All Books</li>

                        <li>Most Recent</li>

                        <li>Most Popular</li>

                        <li>Free Books</li>

                    </ul>

                    <div action="" class="search">

                        <input type="text" name="" id="search">

                        <i class="fa fa-search"></i>

                    </div>

                </div>

                <div id="booklist" class="book-list">

                </div>

            </div>

        </div>

        <div class="foot">

            <div>

                <div class="sett-b"><i></i></div>

                <div class="ask-b"><i></i></div>

            </div>

        </div>

    </div>

    <script type="module" src="./Js/HomePage.js"></script>

</body>

</html>

1. details.html

ملف صفحة تفاصيل الكتب و يحوي التالي

* + <div id="container"> يعرف عنصرًا يحتوي على بطاقات تفاصيل الكتب التي ستتم إضافتها باستخدام جافا سكريبت.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bookstore</title>

</head>

<body>

    <div id="container">

        <!-- JavaScript Generated Books Detais Card-->

    </div>

   <script type="module" src="./Js/details.js"></script>

</body>

</html>

1. wishlist.html

ملف صفحة التفضيلات و تحتوي

* + <h1 >My Wishlist</h1> يعرف هذا العنصر عنوان قائمة التفضيلات.
  + <div id="container" style="display: flex; flex-direction: row; justify-content: space-around;">

يعرف عنصرا يحتوي على بطاقات الكتب المولدة باستخدام جافا سكريبت

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bookstore</title>

</head>

<body>

    <div style="display: flex; align-items: center; justify-content: center;" class="mywishlist">

        <h1 >My Wishlist</h1>

    </div>

    <div id="container" style="display: flex; flex-direction: row; justify-content: space-around;">

 <!-- JavaScript Generated Wishlist's Books Cards-->

    </div>

    <script type="module" src="./Js/wishlist.js"></script>

</body>

</html>

1. Css File:

يحتوى هذا الملف على التنسيقات الاساسية للصفحات السابقة وهي :

1. all.min.css : for font awesome lib.
2. Normalize.css :
   * يتكون ملف normalize.css من مجموعة من القواعد والتعليمات التي تعمل على توحيد الأنماط في المتصفحات المختلفة. يتم تنظيم القواعد في فئات مختلفة مثل "Document" و "Sections" و "Grouping content" و "Text-level semantics" و "Embedded content" و "Forms" و "Interactive" و "Misc".
3. style.css :
   * تعريف المتغيرات المخصصة باستخدام :root.
   * تعريف الأنماط العامة للعناصر مثل li و .container.
   * تعريف الأنماط لعناصر الواجهة العلوية .top.
   * تعريف الأنماط لعناصر الواجهة الوسطى .mid.
   * تعريف الأنماط لعناصر الواجهة الجانبية .side.
   * تعريف الأنماط لعناصر الواجهة السفلية .foot.
4. Img File: Contains all all the images that is used in the project.
5. Webfonts: for fontawesome lib.
6. Signup file: Contains Sign up/in forms and styles:
7. Index.html:
   * <div class="container"> ينشئ عنصرًا تحتوي على جميع عناصر الصفحة.
   * <div class="frame"> ينشئ إطارًا لعناصر الصفحة.
   * <div class="nav"> يحتوي على قائمة التنقل بين صفحة تسجيل الدخول وصفحة التسجيل.
   * <li class="signin-active"> ينشئ عنصرًا في القائمة لصفحة تسجيل الدخول.
   * <a id="signin"> ينشئ رابطًا لصفحة تسجيل الدخول.
   * <li class="signup-inactive"> ينشئ عنصرًا في القائمة لصفحة التسجيل.
   * <a id="signup"> ينشئ رابطًا لصفحة التسجيل.
   * <form class="form-signin"> ينشئ نموذج تسجيل الدخول.
   * <label for="username"> ينشئ تسمية لحقل اسم المستخدم.
   * <input class="form-styling" type="text" name="username"> ينشئ حقل إدخال لاسم المستخدم.
   * <label for="password"> ينشئ تسمية لحقل كلمة المرور.
   * <input class="form-styling" type="text" name="password"> ينشئ حقل إدخال لكلمة المرور.
   * <input type="checkbox" id="checkbox"> ينشئ خانة اختيار للبقاء مسجل الدخول.
   * <label for="checkbox"> ينشئ تسمية لخانة الاختيار.
   * <a id="Signin" class="btn-signin"> ينشئ زر تسجيل الدخول.
   * <p class="err-msg"> يعرض رسالة الخطأ (إذا وجدت).
   * <form class="form-signup"> ينشئ نموذج التسجيل.
   * <label for="fullname"> ينشئ تسمية لحقل الاسم الكامل.
   * <input class="form-styling" type="text" name="fullname"> ينشئ حقل إدخال للاسم الكامل.
   * <label for="email"> ينشئ تسمية لحقل البريد الإلكتروني.
   * <input class="form-styling" type="text" name="email"> ينشئ حقل إدخال للبريد الإلكتروني.
   * <label for="password"> ينشئ تسمية لحقل كلمة المرور.
   * <input class="form-styling" type="text" name="password"> ينشئ حقل إدخال لكلمة المرور.
   * <label for="confirmpassword"> ينشئ تسمية لحقل تأكيد كلمة المرور.
   * <input class="form-styling" type="text" name="confirmpassword"> ينشئ حقل إدخال لتأكيد كلمة المرور.
   * <a id="Signup" class="btn-signup"> ينشئ زر التسجيل.
   * <p class="err-msg"> يعرض رسالة الخطأ (إذا وجدت).

<!DOCTYPE html>

<html lang="en" >

<head>

  <meta charset="UTF-8">

  <title>Book Store</title>

  <link rel="stylesheet" href="./style.css">

</head>

<body>

<div class="container">

  <div class="frame">

    <div class="nav">

      <ul>

        <li class="signin-active"><a id="signin"  >Sign in</a></li>

        <li class="signup-inactive"><a id="signup" >Sign up </a></li>

      </ul>

    </div>

    <div>

          <form class="form-signin" id="form-signin" action="" method="post" name="form">

          <label for="username">Username</label>

          <input class="form-styling" type="text" name="username" placeholder=""/>

          <label for="password">Password</label>

          <input class="form-styling" type="text" name="password" placeholder=""/>

          <input type="checkbox" id="checkbox"/>

          <label for="checkbox" ><span class="ui"></span>Keep me signed in</label>

          <div class="btn-animate">

            <a id="Signin" class="btn-signin">Sign in</a>

          </div>

          <p class="err-msg"></p>

                </form>

                <form class="form-signup" action="" method="post" name="form">

          <label for="fullname">Full name</label>

          <input class="form-styling" type="text" name="fullname" placeholder=""/>

          <label for="email">Email</label>

          <input class="form-styling" type="text" name="email" placeholder=""/>

          <label for="password">Password</label>

          <input class="form-styling" type="text" name="password" placeholder=""/>

          <label for="confirmpassword">Confirm password</label>

          <input class="form-styling" type="text" name="confirmpassword" placeholder=""/>

          <a id="Signup" class="btn-signup">Sign Up</a>

          <p class="err-msg"></p>

                </form>

  </div>

</div>

<script type="module" src="./script.js"></script>

</body>

</html>

1. Style.css: Styles for index.html>
2. Script.js :

يحتوي هذا الملف على :

* + استيراد الكود: يتم استيراد كائن cookie من ملف cookies.js.

الدوال:

* + signupin() لتبديل بين نموذجي تسجيل الدخول والتسجيل.
  + Signup() لتسجيل المستخدم في قاعدة البيانات.
  + Signin() لتسجيل الدخول للمستخدم.
  + send\_signup() لإرسال البيانات واستلام الاستجابة لعملية التسجيل.
  + send\_signin() لإرسال البيانات واستلام الاستجابة لعملية تسجيل الدخول.
  + إضافة الأحداث للأزرار يتم إضافة الأحداث لجميع الأزرار في الواجهة.

import cookie from "../Js/cookies.js";

var cook=new cookie();

//To switch between signin and signup forms

function signupin(){

    document.querySelector('.frame').classList.toggle("frame-long");

    document.querySelector('.form-signin').classList.toggle("form-signin-left");

    document.querySelector('.signup-inactive').classList.toggle("signup-active");

    document.querySelector('.signin-active').classList.toggle("signin-inactive");

    document.querySelector('.form-signup').classList.toggle("form-signup-left");

  }

//To Register a user in database

  let Signup=async ()=>{

    //take fields values

    const feilds=document.querySelector(".form-signup").children;

    const data=[feilds[1].value,feilds[3].value,feilds[5].value,feilds[7].value];

    //Check if empty

    if(data[0]==""||data[1]==""||data[2]==""||data[3]==""){

      let errmsg=document.querySelector('.form-signup').lastElementChild;

          errmsg.innerText="Empty Fields!";

    }else{

      let formdata=new FormData();

      formdata.set("fullname",data[0]);

      formdata.set("email",data[1]);

      formdata.set("password",data[2]);

      formdata.set("confirmpassword",data[3]);

      //get a response after sending the data

      let response= async ()=>{

        //send the data using api

        const a = await send\_signup(formdata);

        console.log(a);

        if(a.status=="Success")

        {

          cook.setCookie("User\_ID",a.User\_ID,365);

          cook.setCookie("User\_Name",a.User\_Name,365);

         window.location.assign(`../index.html?ID=${a.User\_ID}&username=${a.User\_Name}`);

        }

        if(a.status=="failed"){

          alert("Error : "+a.Error+" !");

          return null;

        }

      };

      response();

    }

  }

  //To Login a user

  let Signin=async ()=>{

    //take fields values

    const feilds=document.querySelector(".form-signin").children;

    const data=[feilds[1].value,feilds[3].value];

    //Check if empty

  if(data[0]==""||data[1]==""){

    let errmsg=document.querySelector('.form-signin').lastElementChild;

            errmsg.innerText="Empty Fields!";

  }else{

    let formdata=new FormData();

      formdata.set("username",data[0]);

      formdata.set("password",data[1]);

       //get a response after sending the data

      let response= async ()=>{

        //send the data using api

        const a = await send\_signin(formdata);

        console.log(a);

        if(a.status=="Success")

        {

          cook.setCookie("User\_ID",a.User\_ID,365);

          cook.setCookie("User\_Name",a.User\_Name,365);

          window.location.assign(`../index.html?ID=${a.User\_ID}&username=${a.User\_Name}`);

        }

        if(a.status=="failed"){

          alert("Error : "+a.Error+" !");

          return null;

        }

      };

      response();

  }

  }

  //function to send the data and return a response

  async function send\_signup(formdata){

      let response;

      await fetch("http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/signup.php",

        {method:'POST',body:formdata})

        .then(res=>res.json()).then(data=>response=data[0]);

        return response;

  }

   //function to send the data and return a response

  async function send\_signin(formdata){

    let response;

    await fetch("http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/login.php",

    {method:'POST',body:formdata})

    .then(res=>res.json()).then(data=>response=data[0]);

    return response;

}

//add events to all buttons

document.querySelector('#Signin').addEventListener('click',Signin);

document.querySelector('#Signup').addEventListener('click',Signup);

document.querySelector('#signin').addEventListener('click',signupin);

document.querySelector('#signup').addEventListener('click',signupin);

export  {signupin,Signin,Signup};

1. JS file:

يحتوي هذا المجلد على ملفات جافا سكريبت للتحكم في المقع و عرض البيانات و يحوي الملفات التالية:

* 1. Components.js:

الصف Components يحتوي على وظائف لإنشاء عناصر واجهة المستخدم المختلفةو هي:

* + وظيفة card تقوم بإنشاء بطاقة لعرض معلومات الكتاب بما في ذلك العنوان والسعر والصورة.
  + وظيفة login تقوم بإنشاء زر تسجيل الدخول.
  + وظيفة logout تقوم بإنشاء زر تسجيل الخروج.
  + وظيفة bookDetails تقوم بإنشاء بطاقة لعرض تفاصيل الكتاب بما في ذلك العنوان والمؤلف ورقم ISBN والفئة والوصف والسعر والصورة.

export default class Components{

  //Books Cards

    card(id,title,price,img){

        return `

        <div class="card"">

<div class="image""><img src="./img/${img}" class="img" alt="img" /></div>

  <span class="booktitle">${title}</span>

  <span class="price"">${price}$</span>

  <a class="desc" href="./details.html?id=${id}">Show Details</a>

</div>`;

    }

//Login Button

    login(){

        return (`<button id="login" class="Btn">

        <div class="sign"><svg viewBox="0 0 512 512"><path d="M217.9 105.9L340.7 228.7c7.2 7.2 11.3 17.1 11.3 27.3s-4.1 20.1-11.3 27.3L217.9 406.1c-6.4 6.4-15 9.9-24 9.9c-18.7 0-33.9-15.2-33.9-33.9l0-62.1L32 320c-17.7 0-32-14.3-32-32l0-64c0-17.7 14.3-32 32-32l128 0 0-62.1c0-18.7 15.2-33.9 33.9-33.9c9 0 17.6 3.6 24 9.9zM352 416l64 0c17.7 0 32-14.3 32-32l0-256c0-17.7-14.3-32-32-32l-64 0c-17.7 0-32-14.3-32-32s14.3-32 32-32l64 0c53 0 96 43 96 96l0 256c0 53-43 96-96 96l-64 0c-17.7 0-32-14.3-32-32s14.3-32 32-32z"></path></svg></div>

        <div class="text">Login</div>

      </button>

      `);

    }

//Logout Button

    logout(){

      return ( `

    <button id="logout" class="Btn">

  <div class="sign"><svg viewBox="0 0 512 512"><path d="M377.9 105.9L500.7 228.7c7.2 7.2 11.3 17.1 11.3 27.3s-4.1 20.1-11.3 27.3L377.9 406.1c-6.4 6.4-15 9.9-24 9.9c-18.7 0-33.9-15.2-33.9-33.9l0-62.1-128 0c-17.7 0-32-14.3-32-32l0-64c0-17.7 14.3-32 32-32l128 0 0-62.1c0-18.7 15.2-33.9 33.9-33.9c9 0 17.6 3.6 24 9.9zM160 96L96 96c-17.7 0-32 14.3-32 32l0 256c0 17.7 14.3 32 32 32l64 0c17.7 0 32 14.3 32 32s-14.3 32-32 32l-64 0c-53 0-96-43-96-96L0 128C0 75 43 32 96 32l64 0c17.7 0 32 14.3 32 32s-14.3 32-32 32z"></path></svg></div>

  <div class="text">Logout</div>

</button>

    `);

    }

//Books Details Card

    bookDetails(title,Author,isbn,category,desc,price,img){

      return(`<div class="card">

      <nav>

        <svg class="arrow" version="1.1" viewBox="0 0 512 512" width="512px" xml:space="preserve" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"><polygon points="352,115.4 331.3,96 160,256 331.3,416 352,396.7 201.5,256 " stroke="#727272"/></svg>

        Back to all Plants

        <svg class="heart" version="1.1" viewBox="0 0 512 512" width="512px" xml:space="preserve" stroke="#727272" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"><path d="M340.8,98.4c50.7,0,91.9,41.3,91.9,92.3c0,26.2-10.9,49.8-28.3,66.6L256,407.1L105,254.6c-15.8-16.6-25.6-39.1-25.6-63.9  c0-51,41.1-92.3,91.9-92.3c38.2,0,70.9,23.4,84.8,56.8C269.8,121.9,302.6,98.4,340.8,98.4 M340.8,83C307,83,276,98.8,256,124.8  c-20-26-51-41.8-84.8-41.8C112.1,83,64,131.3,64,190.7c0,27.9,10.6,54.4,29.9,74.6L245.1,418l10.9,11l10.9-11l148.3-149.8  c21-20.3,32.8-47.9,32.8-77.5C448,131.3,399.9,83,340.8,83L340.8,83z" stroke="#727272"/></svg>

      </nav>

      <div class="photo">

        <img src="./img/${img}">

      </div>

      <div class="description">

        <h2>${title}</h2>

        <h4>${Author}</h4>

        <br>

        <h4>${category}</h4>

        <h1>${price}</h1>

        <p>${desc}</p>

        <br>

        <p >ISBN: ${isbn}</p>

        <br>

        <button>Add to Cart</button>

        <button id="wish">Wishlist</button>

      </div>

    </div>`);

    }

}

* 1. Cookies.js:

الصف cookie يحتوي على وظائف لإنشاء و ادارة الكوكي و هي:

* + setCookieستخدم لإنشاء كوكي جديد.
  + getCookieتستخدم للحصول على قيمة الكوكي.
  + checkCookie تستخدم للتحقق مما إذا كان الكوكي موجودًا أم لا.

export default class cookie{

  //to create new cookies

    setCookie(cname, cvalue, exdays) {

        const d = new Date();

        d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

        let expires = "expires="+ d.toUTCString();

        document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

      }

      //to get a cookie's value

     getCookie(cname) {

        let name = cname + "=";

        let decodedCookie = decodeURIComponent(document.cookie);

        let ca = decodedCookie.split(';');

        for(let i = 0; i <ca.length; i++) {

          let c = ca[i];

          while (c.charAt(0) == ' ') {

            c = c.substring(1);

          }

          if (c.indexOf(name) == 0) {

            return c.substring(name.length, c.length);

          }

        }

        return "";

      }

      //check if a cookie exsists

      checkCookie(usr,id) {

        let username = this.getCookie(usr);

        let userid = this.getCookie(id);

        if (username != ""&&userid!="") {

         return true;

        } else {

          return false;

        }

      }

}

* 1. style.js:

يحتوي الملف على صف يسمى Style تحتوي وظائف تعيدتنسيقات للناصر المولدة بالجافا سكربت و هي:

"bookList ، "logout"، و "login". كل واحدة من هذه الوظائف تعيد سلسلة نصية تحتوي على تنسيقات CSS

export default class Style{

    bookList(){

        return (`<style>

        .mid .main .book-list{

            display: flex;

            justify-content: space-evenly;

            margin: 2rem;

        }

        .card{

              width: 11.875em;  height: 20.875em; box-shadow: 0px 1px 13px rgba(0,0,0,0.1); cursor: pointer; transition: all 120ms; display: flex; flex-direction: column;  align-items: center;  justify-content: center;  background: #fff;  padding: 0.5em; ;

        }

        .desc {

          text-decoration: none;

          align-items: center;

          justify-content: center;

          background: #00AC7C;

          color: #fff;

          height: 2.5em;

          width: 90%;

          transition: all 80ms;

          font-weight: 600;

          text-transform: uppercase;

          opacity: 1;

          display: flex;

          }

        .card .price {

            font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;

            font-size: 0.9em;

            left: 0.625em;

            bottom: 0.625em;

            color: #000;

          }

         .card:active {

           transform: scale(0.98);

        }

        .card:active .desc {

           height: 3.125em;

        }

    .booktitle{

        font-family: Arial, Helvetica, sans-serif; font-size: 0.9em;   left: 0.625em; bottom: 1.875em; font-weight: 400; color: #000;

    }

    .price{

        font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;font-size: 0.9em; left: 0.625em; bottom: 0.625em; color: #000;

    }

    .image{

        background: rgb(241, 241, 241); width: 100%; height: 70%; margin:5px;  place-items: center;

    }

    .img{

        width:100%; height: 100%;

    }

       .text {

           max-width: 55px;

        }

     </style>`);

    }

    logout(){

        return (`<style>

        .Btn {

          display: flex;

          align-items: center;

          justify-content: flex-start;

          width: 45px;

          height: 45px;

          border: none;

          border-radius: 50%;

          cursor: pointer;

          position: relative;

          overflow: hidden;

          transition-duration: .3s;

          box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.199);

          background-color: rgb(255, 65, 65);

        }

        /\* plus sign \*/

        .sign {

          width: 100%;

          transition-duration: .3s;

          display: flex;

          align-items: center;

          justify-content: center;

        }

        .sign svg {

          width: 17px;

        }

        .sign svg path {

          fill: white;

        }

        /\* text \*/

        .text {

          position: absolute;

          right: 0%;

          width: 0%;

          opacity: 0;

          color: white;

          font-size: 1.2em;

          font-weight: 600;

          transition-duration: .3s;

        }

        /\* hover effect on button width \*/

        .Btn:hover {

          width: 125px;

          border-radius: 40px;

          transition-duration: .3s;

        }

        .Btn:hover .sign {

          width: 30%;

          transition-duration: .3s;

          padding-left: 20px;

        }

        /\* hover effect button's text \*/

        .Btn:hover .text {

          opacity: 1;

          width: 70%;

          transition-duration: .3s;

          padding-right: 10px;

        }

        /\* button click effect\*/

        .Btn:active {

          transform: translate(2px ,2px);

        }

        </style>`);

    }

    login(){

        return (`<style>

        .Btn {

          display: flex;

          align-items: center;

          justify-content: flex-start;

          width: 45px;

          height: 45px;

          border: none;

          border-radius: 50%;

          cursor: pointer;

          position: relative;

          overflow: hidden;

          transition-duration: .3s;

          box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.199);

          background-color: rgb(163, 142, 255);

        }

        /\* plus sign \*/

        .sign {

          width: 100%;

          transition-duration: .3s;

          display: flex;

          align-items: center;

          justify-content: center;

        }

        .sign svg {

          width: 17px;

        }

        .sign svg path {

          fill: white;

        }

        /\* text \*/

        .text {

          position: absolute;

          right: 0%;

          width: 0%;

          opacity: 0;

          color: white;

          font-size: 1.2em;

          font-weight: 600;

          transition-duration: .3s;

        }

        /\* hover effect on button width \*/

        .Btn:hover {

          width: 125px;

          border-radius: 40px;

          transition-duration: .3s;

        }

        .Btn:hover .sign {

          width: 30%;

          transition-duration: .3s;

          padding-left: 20px;

        }

        /\* hover effect button's text \*/

        .Btn:hover .text {

          opacity: 1;

          width: 70%;

          transition-duration: .3s;

          padding-right: 10px;

        }

        /\* button click effect\*/

        .Btn:active {

          transform: translate(2px ,2px);

        }

        </style>`);

    }

    bookDetails(){

      return(`<style>

      @import url(https://fonts.googleapis.com/css?family=Raleway:400,300,500,700);

 \* {

   box-sizing: border-box;

}

 body {

   background: #aedaa6;

   font-family: "Raleway";

}

 body .card {

   width: 650px;

   position: absolute;

   background: white;

   margin: 0 auto;

   top: 50%;

   left: 50%;

   transform: translate(-50%, -50%);

   box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 2px 10px 0 rgba(0, 0, 0, 0.12);

   transition: all 0.3s;

}

 body .card:hover {

   box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

}

 body .card nav {

   width: 100%;

   color: #727272;

   text-transform: uppercase;

   padding: 20px;

   border-bottom: 2px solid #efefef;

   font-size: 12px;

}

 body .card nav svg.heart {

   height: 24px;

   width: 24px;

   float: right;

   margin-top: -3px;

   transition: all 0.3s ease;

   cursor: pointer;

}

 body .card nav svg.heart:hover {

   fill: red;

}

 body .card nav svg.arrow {

   float: left;

   height: 15px;

   width: 15px;

   margin-right: 10px;

}

 body .card .photo {

   padding: 30px;

   width: 45%;

   text-align: center;

   float: left;

}

 body .card .photo img {

   max-height: 240px;

}

 body .card .description {

   padding: 30px;

   float: left;

   width: 55%;

   border-left: 2px solid #efefef;

}

 body .card .description h1 {

   color: #515151;

   font-weight: 300;

   padding-top: 15px;

   margin: 0;

   font-size: 30px;

   font-weight: 300;

}

 body .card .description h2 {

   color: #515151;

   margin: 0;

   text-transform: uppercase;

   font-weight: 500;

}

 body .card .description h4 {

   margin: 0;

   color: #727272;

   text-transform: uppercase;

   font-weight: 500;

   font-size: 12px;

}

 body .card .description p {

   font-size: 12px;

   line-height: 20px;

   color: #727272;

   padding: 20px 0;

   margin: 0;

}

 body .card .description button {

   outline: 0;

   border: 0;

   background: none;

   border: 1px solid #d9d9d9;

   padding: 8px 0px;

   margin-bottom: 30px;

   color: #515151;

   text-transform: uppercase;

   width: 125px;

   font-family: inherit;

   margin-right: 5px;

   transition: all 0.3s ease;

   font-weight: 500;

}

 body .card .description button:hover {

   border: 1px solid #aedaa6;

   color: #aedaa6;

   cursor: pointer;

}

      </style>`);

    }

}

* 1. homepage.js:

الملف يحوي على الوظائف التالية:

* + Book هذه الوظيفة تستخدم لجلب بيانات الكتب من قاعدة البيانات وعرضها على الصفحة الرئيسية. تستخدم XMLHttpRequest لجلب البيانات وتقوم بتطبيق عامل تصفية باستخدام شريط البحث. ثم تقوم بإنشاء بطاقة لكل كتاب وإضافتها إلى الصفحة.
  + ISIN هذه الوظيفة تستخدم للتحقق مما إذا كان المستخدم قد قام بتسجيل الدخول أم لا. إذا كان المستخدم قد قام بتسجيل الدخول، فإنها تقوم بإزالة زر تسجيل الدخول / التسجيل وتضيف زر تسجيل الخروج. إذا لم يكن المستخدم قد قام بتسجيل الدخول، فإنها تقوم بإزالة زر تسجيل الخروج وتضيف زر تسجيل الدخول.
  + logout هذه الوظيفة تستخدم لتسجيل خروج المستخدم من الموقع. تقوم بحذف ملفات الكوكيز المتعلقة بتسجيل الدخول وإعادة تحميل الصفحة.
  + login هذه الوظيفة تستخدم لإرسال المستخدم إلى صفحة التسجيل.
  + ToWishlist هذه الوظيفة تستخدم لإرسال المستخدم إلى صفحة قائمة الرغبات الخاصة به إذا كان قد قام بتسجيل الدخول. إذا لم يكن المستخدم قد قام بتسجيل الدخول، فإنها تعرض رسالة تنبيه تفيد بأنه يجب تسجيل الدخول أولاً.

/\*\*

 \* This file is used to add elements to main index page

 \* elements that need data from the backend

 \* and handle sign in/up status

 \* \*\*/

import cookie from "./cookies.js";

import Style from "./style.js";

import Components from "./components.js";

var cook=new cookie();

var styles=new Style();

var comp=new Components();

//This function is used to fetch books data from the database

let Book=async()=>{

    var array;

    //implay a filter to the fetched data using search bar

    var title=document.getElementById('search').value;

    document.getElementById("booklist").innerHTML=null;

    document.getElementById("booklist").innerHTML+=styles.bookList();

    console.log(title);

    //start fetching data

    let req=new XMLHttpRequest();

    req.onreadystatechange=()=>{

        if(req.readyState===4 && req.status===200){

            let res=JSON.parse(req.responseText);

            console.log(res);

            array=res;

        }

    };

     req.open("GET",`http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/books.php?title=${title}`,false);

     req.send();

     //Generate Cards for each book

    array.forEach(element => {

        document.getElementById("booklist").innerHTML+=comp.card(element["id"],element["title"],element["price"],element["img"]);

    });

}

//This function is used to check if the user is signed In and add signup/in button

let ISIN=()=>{

  let flag=cook.checkCookie("User\_Name","User\_ID");

  if(flag){

    document.getElementById("r-status").innerHTML=null;

    document.getElementById("r-status").innerHTML+=styles.logout();

    document.getElementById("r-status").innerHTML+=comp.logout();

    document.getElementById('logout').addEventListener('click',logout);

  }else{

    document.getElementById("r-status").innerHTML=null;

    document.getElementById("r-status").innerHTML+=styles.login();

    document.getElementById("r-status").innerHTML+=comp.login();

  document.getElementById('login').addEventListener('click',login);

  }

};

//This is used to logout the user from the website

let logout=()=>{

  cook.setCookie("User\_ID","",-1);

  cook.setCookie("User\_Name","",-1);

  window.location.reload();

}

//this is used to send the user to registration form

let login=()=>{

  window.location.assign("./Signup/index.html");

}

//this is used to send the user to his wishlist if signedin

let ToWishlist=()=>{

  let flag=cook.checkCookie("User\_Name","User\_ID");

  console.log(flag);

  if(flag){

   let id=cook.getCookie("User\_ID");

   window.location.assign(`./wishlist.html`);

  }else{

    alert("You are Not Signed In!!")

  }

}

window.addEventListener('load',ISRegister);

window.addEventListener('load',Book);

document.getElementById('search').addEventListener('keyup',Book);

document.getElementById("wishlist").addEventListener("click",ToWishlist);

export {Book,ISIN,logout,ToWishlist};

* 1. wishlist.js:

الملف يحوي على الوظائف التالية:

* + wishlist التي تقوم بجلب بيانات قائمة الرغبات للمستخدم وإنشاء بطاقات لكل عنصر في القائمة.

import cookie from "./cookies.js";

import Style from "./style.js";

import Components from "./components.js";

var cook=new cookie();

var styles=new Style();

var comp=new Components();

//function to get data of the user's wshlist an generate cards for it

let wishlist=async()=>{

    var array;

    var id= cook.getCookie("User\_ID");

    console.log(id);

    document.getElementById("container").innerHTML=null;

    document.getElementById("container").innerHTML+=styles.bookList();

    console.log(id);

    //get the data

    let req=new XMLHttpRequest();

    req.onreadystatechange=()=>{

        if(req.readyState===4 && req.status===200){

            let res=JSON.parse(req.responseText);

            console.log(res);

            array=res;

        }

    };

     req.open("GET",`http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/wishlist.php?user\_id=${id}`,false);

     req.send();

    array.forEach(element => {

        document.getElementById("container").innerHTML+=comp.card(element["id"],element["title"],element["price"],element["img"]);

    });

}

window.addEventListener('load',wishlist);

export default {wishlist}

* 1. details.js:

1. الملف يحوي على الوظائف التالية:
   * "Book" للحصول على تفاصيل الكتاب.
   * "addToWishlist" لإضافة الكتاب إلى قائمة رغبات المستخدم.

import Components from "./components.js";

import Style from "./style.js";

import cookie from "./cookies.js";

 var com=new Components();

 var style=new Style();

 var cook=new cookie();

 //Get Books Details

 let Book=()=>{

    var book;

    const id=location.search.substring(1).split('=').slice(-1)[0];

    console.log(id);

   document.getElementById("container").innerHTML=null;

    document.getElementById("container").innerHTML+=style.bookDetails();

    let req=new XMLHttpRequest();

    req.onreadystatechange=()=>{

        if(req.readyState===4 && req.status===200){

            let res=JSON.parse(req.responseText);

            console.log(res);

            book=res;

        }

    };

     req.open("GET",`http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/books.php?id=${id}`,false);

     req.send();

    document.getElementById("container").innerHTML+=com.bookDetails(

        book[0].title,

        book[0].author,

        book[0].ISBN,

        book[0].category,

        book[0].description,

        book[0].price,

        book[0].img);

        document.getElementById("wish").addEventListener("click",addToWishlist);

 }

 //add the book to the user's wishlist

 let addToWishlist=()=>{

let flag=cook.checkCookie("User\_Name","User\_ID");

  console.log(flag);

  if(flag){

    var array;

    var user\_id= cook.getCookie("User\_ID");

    var book\_id=location.search.substring(1).split('=').slice(-1)[0];

    let req=new XMLHttpRequest();

    req.onreadystatechange=()=>{

        if(req.readyState===4 && req.status===200){

            let res=JSON.parse(req.responseText);

            console.log(res);

            array=res;

        }

    };

     req.open("GET",`http://127.0.0.1/Yazan\_174681\_BWP\_S23\_HW/php/api/wishlist.php?user\_id=${user\_id}&&book\_id=${book\_id}`,false);

     req.send();

if(array==="success"){

    alert("Book is added to yor wishlist");

}else{

    alert("Book is exsists");

}

  }else{

    alert("You are Not Signed In!!")

    window.location.assign("./Signup/index.html");

  }

 }

 window.addEventListener('load',Book);

export default {Book,addToWishlist};

1. PHP file:
   * 1. Api file:
        1. Book.php:

يتكون الملف من الأجزاء التالية:

* + التحقق من طريقة الطلب: يتم التحقق من أن الطلب المرسل إلى الخادم هو GET.
  + إنشاء كائنات الكتب: يتم إنشاء كائنات الكتب باستخدام معلومات البحث المرسلة في الطلب. إذا تم توفير معلومات العنوان (title)، يتم إنشاء كائن booksContr باستخدام العنوان كمعلمة. إذا تم توفير معلومات الهوية (id)، يتم إنشاء كائن booksContr باستخدام الهوية كمعلمة.
  + استرداد البيانات: يتم استدعاء الدالة books() في كائن booksContr لاسترداد بيانات الكتب من قاعدة البيانات.
  + إرجاع البيانات: يتم إرجاع بيانات الكتب المسترجعة في تنسيق JSON باستخدام دالة json\_encode().
  + التحقق من طريقة الطلب: إذا لم يكن الطلب من نوع GET، يتم إرجاع رسالة توضح أن الطلب يجب أن يكون من نوع GET.

<?php

header('Access-Control-Allow-Origin:\*');

header('Content-Type:application/json');

if($\_SERVER["REQUEST\_METHOD"]==="GET"){

     //Instantiate book classes

     include "../model/dbh.model.php";

     include "../model/books.model.php";

     include "../controller/books.contr.php";

    //grabbing the data

    if(isset($\_GET["title"]))

   {

    $books=new booksContr($\_GET["title"],"title");

   }elseif(isset($\_GET["id"])){

    $books=new booksContr($\_GET["id"],"id");

   }

    //Running return the data in JSON Encode

    echo json\_encode($books->books());

}else{

    echo json\_encode("Should be GET Request");

}

* + - 1. wishlist.php:

يتكون الملف من الأجزاء التالية:

* + التحقق من طريقة الطلب: يتم التحقق من أن الطلب المرسل هو طلب GET.
  + إنشاء كائن WishlistContr: يتم إنشاء كائن من الفئة WishlistContr وتمرير معرف المستخدم كمعلمة.
  + إضافة الكتاب إلى قائمة الرغبات: إذا تم توفير معرف الكتاب ومعرف المستخدم في الطلب، يتم استدعاء دالة addbook لإضافة الكتاب إلى قائمة الرغبات.
  + عرض الكتب في قائمة الرغبات: إذا تم توفير معرف المستخدم فقط في الطلب، يتم استدعاء دالة showbooks لعرض الكتب الموجودة في قائمة الرغبات.
  + الاستجابة بصيغة JSON: يتم تحويل النتيجة إلى صيغة JSON وإرجاعها كاستجابة.

1. <?php
2. header('Access-Control-Allow-Origin:\*');
3. header('Content-Type:application/json');
4. if($\_SERVER["REQUEST\_METHOD"]==="GET"){
5. //Instantiate wishlist classes
6. include "../model/dbh.model.php";
7. include "../model/wishlist.model.php";
8. include "../controller/wishlist.contr.php";
9. //grabbing the data
10. $wishlist=new WishlistContr($\_GET["user\_id"]);
11. if(isset($\_GET["user\_id"])&&isset($\_GET["book\_id"])){
12. echo json\_encode($wishlist->addbook($\_GET["book\_id"]));
13. }elseif(isset($\_GET["user\_id"])){
15. echo json\_encode($wishlist->showbooks());
16. }
18. }else{
19. echo json\_encode("Should be GET Request");
20. }
    * + 1. login.php:

يقوم هذا الملف بالتالي :

* + نقوم بفحص طريقة الطلب، إذا كانت POST ، نقوم بمعالجة بيانات تسجيل الدخول المرسلة من المستخدم.
  + نقوم بإنشاء كائن من الصف LoginContr ونمرر إليه بيانات تسجيل الدخول.
  + نقوم بتشغيل معالج الأخطاء وتسجيل المستخدم باستخدام دالة loginUser()
  + بناءً على النتيجة، نقوم بإرجاع رد مناسب إلى العميل في تنسيق JSON.

<?php

header('Access-Control-Allow-Origin:\*');

header('Content-Type:application/json');

if($\_SERVER["REQUEST\_METHOD"]==="POST"){

    //grabbing the date

    $fullname=$\_POST["username"];

    $password=$\_POST["password"];

    //Instantiate signincontr classes

    include "../model/dbh.model.php";

    include "../model/login.model.php";

    include "../controller/login.contr.php";

    $login=new LoginContr($fullname,$password);

    //Running erroe handlers and user sign up

    switch($login->loginUser())

    {

        case "ERR:Empty Field":

            echo json\_encode(array(["status"=>"failed","Error"=>"Empty Field"]));

            break;

        case "ERR:Invalid Username":

            echo json\_encode(array(["status"=>"failed","Error"=>"Invalid Username"]));

            break;

        case "Unkown ERR":

            echo json\_encode(array(["status"=>"failed","Error"=>"User Or Password is Wrong!"]));

            break;

        case "Success":

            echo json\_encode(array(["status"=>"Success",

            "User\_ID"=>$\_SESSION['User\_ID'],

            "User\_Name"=>$\_SESSION['User\_Name']]));

            break;

    }

}else{

    echo json\_encode("Should be POST Request");

}

* + - 1. signup.php:

يقوم هذا الملف بالتالي :

* + نقوم بفحص طريقة الطلب، إذا كانت POST ، نقوم بمعالجة بيانات إنشاء الحساب المرسلة من المستخدم.
  + نقوم بإنشاء كائن من الصف SignupContr ونمرر إليه بيانات التسجيل.
  + نقوم بتشغيل معالج الأخطاء وإنشاء المستخدم باستخدام دالة signupUser()
  + نقوم بعد نجاح إنشاء المستخدم بتسجيل دخوله باستخدام الدالة loginUser()
  + بناءً على النتيجة، نقوم بإرجاع رد مناسب إلى العميل في تنسيق JSON.

<?php

header('Access-Control-Allow-Origin:\*');

header('Content-Type:application/json');

if($\_SERVER["REQUEST\_METHOD"]==="POST"){

    //grabbing the date

    $fullname=$\_POST["fullname"];

    $email=$\_POST["email"];

    $password=$\_POST["password"];

    $confirmpassword=$\_POST["confirmpassword"];

    //Instantiate signupcontr classes

    include "../model/dbh.model.php";

    include "../model/signup.model.php";

    include "../model/login.model.php";

    include "../controller/signup.contr.php";

    include "../controller/login.contr.php";

    $signup=new SignupContr($fullname,$email,$password,$confirmpassword);

    //Running erroe handlers and user sign up

   switch ($signup->signupUser())

    {

        case "ERR:Empty Field":

            echo json\_encode(array(["status"=>"failed","Error"=>"Empty Field"]));

            break;

        case "ERR:Invalid Username":

            echo json\_encode(array(["status"=>"failed","Error"=>"Invalid Username"]));

            break;

        case "ERR:Invalid Email":

            echo json\_encode(array(["status"=>"failed","Error"=>"Invalid Email"]));

            break;

        case "ERR:Password Does Not Match":

            echo json\_encode(array(["status"=>"failed","Error"=>"Password Does Not Match"]));

            break;

        case "ERR:Username Is Already Been Taken":

            echo json\_encode(array(["status"=>"failed","Error"=>"Username Is Already Been Taken"]));

            break;

        case "Unkown ERR":

            echo json\_encode(array(["status"=>"failed","Error"=>"Unkown"]));

            break;

        case "Success":

            //Log in the user

            $login=new LoginContr($fullname,$password);

            $login->loginUser();

            echo json\_encode(array(["status"=>"Success",

            "User\_ID"=>$\_SESSION['User\_ID'],

            "User\_Name"=>$\_SESSION['User\_Name']]));

            break;

    }

}else{

    echo json\_encode("Should be POST Request");

}

* + 1. controller file:
       1. login.contr.php:

يحوي هذا الملف التالي :

* + المتغيرات الخاصة $\_fullname و $\_password هذه المتغيرات تحتوي على اسم المستخدم وكلمة المرور المدخلة من قبل المستخدم.
  + الوظيفة \_\_construct() هذه الوظيفة تقوم بتعيين قيم المتغيرات $\_fullname و $\_password عند إنشاء كائن من الصف LoginContr.
  + الوظيفة loginUser() هذه الوظيفة تقوم بتنفيذ عملية تسجيل الدخول. تقوم الوظيفة بفحص ما إذا كانت هناك حقول فارغة أو اسم مستخدم غير صالح، ثم تقوم بإرسال البيانات إلى المودل وإعادة رسالة بنجاح العملية أو فشلها.
  + الوظيفة emptyInput() هذه الوظيفة تقوم بفحص ما إذا كان أحد الحقول فارغة أم لا.
  + الوظيفة invalidFullname() هذه الوظيفة تقوم بفحص ما إذا كان اسم المستخدم صالحًا أم لا.

<?php

class LoginContr extends LoginModel{

    private $\_fullname;

    private $\_password;

    public function \_\_construct(

    $fullname,

    $password)

    {

        $this->\_fullname=$fullname;

        $this->\_password=$password;

    }

    //This function is to login the user

    public function loginUser(){

        //check if any data is empty

        if($this->emptyInput()){

            return "ERR:Empty Field";

        }

        //chaeck if the username is valid

        if($this->invalidFullname()){

            return "ERR:Invalid Username";

        }

        //send the data to model and get the user data if user is exsist

        if(!$this->getUser($this->\_fullname,$this->\_password)){

            return "Success";

        }else{

            return "Unkown ERR";

        }

    }

    //function to check if empty

    private function emptyInput():bool{

        if (empty($this->\_fullname)||empty($this->\_password)) {

            return true;

        }else{

            return false;

        }

    }

    //function to check if user name is valid

    private function invalidFullname():bool

    {

        if(!preg\_match("/^[a-zA-Z0-9]\*$/",$this->\_fullname)){

            return true;

        }

        else{

            return false;

        }

    }

}

* + - 1. signup.contr.php:

يحوي هذا الملف التالي :

* + المتغيرات الخاصة $\_fullname,$\_password ,$\_email, $\_confirm\_passwordهذه المتغيرات تحتوي على اسم المستخدم وكلمة المرور المدخلة من قبل المستخدم.
  + الوظيفة \_\_construct() هذه الوظيفة تقوم بتعيين قيم المتغيرات $\_fullname,$\_password ,$\_email, $\_confirm\_password عند إنشاء كائن من الصف SignupContr.
  + الوظيفة signupUser() هذه الوظيفة تقوم بتنفيذ عملية إنشاء مستخدم جديد. تقوم الوظيفة بفحص ما إذا كانت هناك حقول فارغة أو اسم مستخدم غير صالح أو الايميل غير صالح أو المستخدم موجود مسبقا، ثم تقوم بإرسال البيانات إلى المودل لإنشاء مستخدم جديد.
  + الوظيفة emptyInput() هذه الوظيفة تقوم بفحص ما إذا كان أحد الحقول فارغة أم لا.
  + الوظيفة invalidFullname() هذه الوظيفة تقوم بفحص ما إذا كان اسم المستخدم صالحًا أم لا.
  + الوظيفة invalidEmail() هذه الوظيفة تقوم بفحص ما إذا كان البريد الالكتروني صالحًا أم لا.
  + الوظيفة IsPassowrdMatch() هذه الوظيفة تقوم بفحص ما إذا كانت كلمات المرور متطابقة أم لا.
  + الوظيفة IsUsernameTaken() هذه الوظيفة تقوم بفحص ما إذا كان اسم المسستخدم موجود أم لا.

<?php

class SignupContr extends SignupModel{

    private $\_fullname;

    private $\_password;

    private $\_email;

    private $\_confirm\_password;

    public function \_\_construct(

    $fullname,

    $email,

    $password,

    $confirm\_password)

    {

        $this->\_fullname=$fullname;

        $this->\_email=$email;

        $this->\_password=$password;

        $this->\_confirm\_password=$confirm\_password;

    }

    //This function is to signup the user

    public function signupUser(){

        //check if any field of data is empty

    if($this->emptyInput()){

        return "ERR:Empty Field";

    }

    //check if username is valid

    if($this->invalidFullname()){

        return "ERR:Invalid Username";

    }

    //check if the email is valid

    if($this->invalidEmail()){

        return "ERR:Invalid Email";

    }

    //check if passwords match

    if(!$this->IsPasswordMatch()){

        return "ERR:Password Does Not Match";

    }

    //check if user is already registered

    if($this->IsUsernameTaken()){

        return "ERR:Username Is Already Been Taken";

    }

    if(!$this->setUser($this->\_fullname,$this->\_password,$this->\_email))

    {

        return "Success";

    }else{

        return "Unkown ERR";

    }

    }

    //function to detect the empty data

    private function emptyInput():bool{

        if (empty($this->\_fullname)||empty($this->\_password)||empty($this->\_email)||empty($this->\_confirm\_password)) {

            return true;

        }else{

            return false;

        }

    }

    //function to check if username is valid

    private function invalidFullname():bool

    {

        if(!preg\_match("/^[a-zA-Z0-9]\*$/",$this->\_fullname)){

            return true;

        }

        else{

            return false;

        }

    }

    //function to check if email is valid

    private function invalidEmail():bool

    {

        if(!filter\_var($this->\_email,FILTER\_VALIDATE\_EMAIL)){

            return true;

        }

        else{

            return false;

        }

    }

    //function to check if passwords are match

    private function IsPasswordMatch():bool

    {

        if($this->\_password===$this->\_confirm\_password){

            return true;

        }

        else{

            return false;

        }

    }

    //function to check if user is already registered

    private function IsUsernameTaken():bool

    {

        if($this->checkUser($this->\_fullname,$this->\_email)){

            return true;

        }

        else{

            return false;

        }

    }

}

* + - 1. books.contr.php:

يحوي هذا الملف التالي :

* + المتغيرات الخاصة $\_Input , $\_Typeهذه المتغيرات تحتوي القيم المدخلة من قبل المستخدم.و نوعها (معرف أو عنوان)
  + الوظيفة book()هذه الوظيفة تقوم بارسال البيانات المدخلة من قبل المستخدم إلى المدل و يعيد بيانات عن الكتاب المطلوب من قبل المستخدم.
  + الوظيفة prepare() هذه الوظيفة تقوم بتجزئة النص المدخل إلى كلمات لارسالهافيما بعد للمودل.

<?php

class booksContr extends booksModel{

    private $\_Input;

    private $\_type;

    public function \_\_construct(

    $Input,$type)

    {

        $this->\_Input=$Input;

        $this->\_type=$type;

    }

    public function books(){

        if($this->\_type=="title"){

            //get books by title

            $data=$this->getbooks($this->prepare($this->\_Input));

            return $data;

        }elseif($this->\_type=="id"){

            //get books by id

            $data=$this->getbook($this->prepare($this->\_Input));

            return $data;

        }

    }

//this function is to split the string into words

    private function prepare($string)

    {

       return $array = str\_getcsv($string, ' ', ',',':',);

    }

}

* + - 1. wishlist.contr.php:

يحوي هذا الملف التالي :

* + المتغيرات الخاصة $\_user\_idهذه المتغيرات تحتوي معرف المستخدم.
  + الوظيفة showbooks()هذه الوظيفة تقوم بارسال معرف المستخدم إلى المودل و يعيد الكتاب الموجودة في قائمة رغباته.
  + الوظيفة add($Book\_id) هذه الوظيفة تستقبل باراميتر معرف الككتاب و تستخدم لإضافة الكتاب إلى قائمة رغبات المستخدم.
  + الوظيفة prepare() هذه الوظيفة تقوم بتجزئة النص المدخل إلى كلمات لارسالهافيما بعد للمودل

<?php

class WishlistContr extends WishlistModel{

    private $\_user\_id;

    public function \_\_construct(

    $id)

    {

        $this->\_user\_id=$id;

    }

    public function showbooks(){

        return $data=$this->show($this->prepare($this->\_user\_id));

        }

        public function addbook($book\_id){

           return $data=$this->add($this->\_user\_id,$book\_id);

        }

    private function prepare($string)

    {

       return $array = str\_getcsv($string, ' ', ',',':',);

    }

}

* + 1. model file:
       1. dbh.model.php:

في هذا الملف ، يتم إنشاء كلاس يسمى Dbh. يحتوي هذا الكلاس على وظيفة واحدة تسمى connect. تقوم هذه الوظيفة بإنشاء اتصال بقاعدة البيانات باستخدام PDO. تم تعيين اسم المستخدم وكلمة المرور للاتصال بقاعدة البيانات في المتغيرات $username و $password على التوالي. تم استخدام قيمة localhost للمضيف وقيمة bookstore لاسم قاعدة البيانات. إذا حدث خطأ أثناء الاتصال ، سيتم طباعة رسالة الخطأ وإنهاء البرنامج.

<?php

class Dbh{

    protected function connect()

    {

        try{

            $username='root';

            $password='Yazan112123';

            $dbh=new PDO('mysql:host=localhost;dbname=bookstore',$username,$password);

            return $dbh;

        }catch(PDOException $e)

        {

            print "Error!: ".$e->getMessage()."<br>";

            die();

        }

    }

}

* + - 1. books.model.php:

يحوي هذا الملف التالي :

* + وظيفة "getbooks" تستخدم لاسترداد جميع الكتب من قاعدة البيانات. تقوم الوظيفة بتنفيذ استعلام SQL واسترداد البيانات المطابقة. ثم يتم استدعاء وظيفة "filter" لتصفية البيانات بناءً على عنوان الكتاب المحدد.
  + وظيفة "getbook" تستخدم لاسترداد كتاب واحد بناءً على معرف الكتاب المحدد. تقوم الوظيفة بتنفيذ استعلام SQL واسترداد البيانات المطابقة.
  + وظيفة "filter" تستخدم لتصفية البيانات بناءً على عنوان الكتاب المحدد. تقوم الوظيفة بمقارنة عنوان الكتاب بكلمات البحث المحددة وترجع البيانات المطابقة.

<?php

class booksModel extends Dbh

{

    protected function getbooks($title)

    {

        $stmt=$this->connect()->prepare('SELECT \* FROM books WHERE 1=1;');

        if(!$stmt->execute()){

            $stmt=null;

            return true;

            exit();

        }

        $data=$stmt->fetchAll(PDO::FETCH\_ASSOC);

            return $newData=$this->filter($title,$data);

        $stmt=null;

    }

    protected function getbook($id)

    {

        $stmt=$this->connect()->prepare('SELECT \* FROM books WHERE id=?;');

        if(!$stmt->execute($id)){

            $stmt=null;

            return true;

            exit();

        }

        $data=$stmt->fetchAll(PDO::FETCH\_ASSOC);

        $stmt=null;

        return $data;

    }

    protected function filter($title,$data) {

        $new=[];

        for($i=0;$i<sizeof($data);$i++){

            $flag=true;

            foreach ($title as $str) {

                if (strpos(strtolower($data[$i]['title']),strtolower($str)) === FALSE) {

                    $flag=false;

                    break;

                }

            }

            if($flag){

            array\_push($new,$data[$i]);

            }

        }

       return $new;

    }

    }

* + - 1. wishlist.model.php:

يحوي هذا الملف التالي :

* + الوظيفة show($id) هذه الوظيفة تستخدم لاسترداد الكتب الموجودة في قائمة الرغبات للمستخدم المحدد. تقوم الوظيفة بإعداد استعلام SQL لاسترداد الكتب من جدول الكتب وجدول قائمة الرغبات و يتم تمرير معرف المستخدم كمعامل للاستعلام. إذا فشل تنفيذ الاستعلام، يتم إنهاء التنفيذ..
  + الوظيفة add($user\_id, $book\_id) هذه الوظيفة تستخدم لإضافة كتاب جديد إلى قائمة الرغبات للمستخدم المحدد. قبل إضافة الكتاب، يتم التحقق مما إذا كان الكتاب موجودًا بالفعل في قائمة الرغبات باستخدام الوظيفة checkbook(). إذا لم يكن الكتاب موجودًا، يتم إعداد استعلام SQL لإدخال الكتاب في جدول قائمة الرغبات. إذا فشل تنفيذ الاستعلام، يتم إرجاع "faild". إذا تمت إضافة الكتاب بنجاح، إرجاع "success". إذا كان الكتاب موجودًا بالفعل، يتم إرجاع "exsists".
  + الوظيفة checkbook($user\_id, $book\_id) هذه الوظيفة تستخدم للتحقق مما إذا كان الكتاب موجودًا بالفعل في قائمة الرغبات للمستخدم المحدد. تقوم الوظيفة بإعداد استعلام SQL لاسترداد السجلات من جدول قائمة الرغبات باستخدام معرف الكتاب ومعرف المستخدم كمعاملات للاستعلام. إذا كان هناك سجلات موجودة، يتم إرجاع قيمة صحيحة (true)، وإلا يتم إرجاع قيمة خاطئة (false).

<?php

class WishlistModel extends Dbh

{

    protected function show($id)

{

    $stmt=$this->connect()->prepare('SELECT books.id as id ,books.title as title,books.price as price,books.img as img

    FROM books

    INNER JOIN wishlist ON wishlist.book\_id=books.id WHERE wishlist.user\_id=?;');

    if(!$stmt->execute($id)){

        $stmt=null;

        exit();

    }

    $data=$stmt->fetchAll(PDO::FETCH\_ASSOC);

    $stmt=null;

    return $data;

}

protected function add($user\_id,$book\_id)

{   if(!$this->checkbook($user\_id,$book\_id)){

    $stmt=$this->connect()->prepare('INSERT INTO `wishlist`(`book\_id`, `user\_id`) VALUES (?,?);');

    if(!$stmt->execute(array($book\_id,$user\_id))){

        $stmt=null;

        return "faild";

    }else{

        $stmt=null;

        return "success";

    }

}else{

    return "exsists";

}

}

protected function checkbook($user\_id,$book\_id)

{

    $stmt=$this->connect()->prepare('SELECT book\_id, user\_id From wishlist WHERE book\_id=? and user\_id=?;');

    if(!$stmt->execute(array($book\_id,$user\_id))){

        $stmt=null;

        exit();

    }

    if($stmt->rowCount()>0){

        return true;

    }else{

        return false;

    }

}

}

* + - 1. Signup.model.php:

يحوي هذا الملف التالي :

* + checkUser() هذه الوظيفة تستخدم للتحقق من وجود مستخدم بنفس الاسم أو البريد الإلكتروني في قاعدة البيانات. تستخدم الدالة prepare() لإعداد الاستعلام وتمرر القيم المطلوبة كمعاملات. إذا نجح تنفيذ الاستعلام، فإنه يتم التحقق من عدد الصفوف المتطابقة باستخدام الدالة rowCount() وإرجاع القيمة المناسبة.
  + setUser() هذه الوظيفة تستخدم لإضافة مستخدم جديد إلى قاعدة البيانات. تستخدم الدالة prepare() لإعداد الاستعلام وتمرر القيم المطلوبة كمعاملات. يتم تشفير كلمة المرور باستخدام الدالة password\_hash() قبل إدراجها في قاعدة البيانات.

<?php

class SignupModel extends Dbh

{

    protected function checkUser($fullname,$email)

{

    $stmt=$this->connect()->prepare('SELECT user\_name From users WHERE user\_name=? OR email=?;');

    if(!$stmt->execute(array($fullname,$email))){

        $stmt=null;

        exit();

    }

    if($stmt->rowCount()>0){

        return true;

    }else{

        return false;

    }

}

protected function setUser($fullname,$password,$email)

{

    $stmt=$this->connect()->prepare('INSERT INTO users (user\_name,email,user\_password)VALUES(?,?,?);');

    $hashedpwd=password\_hash($password,PASSWORD\_DEFAULT);

    if(!$stmt->execute(array($fullname,$email,$hashedpwd))){

        $stmt=null;

        exit();

    }

    $stmt=null;

}

}

* + - 1. login.model.php:

يحوي هذا الملف التالي :

وظيفة واحدة getUser() التي تقوم بالتحقق من صحة اسم المستخدم وكلمة المرور وتسجيل الدخول إذا كانت صحيحة و تعمل كالتالي:

* + الاتصال بقاعدة البيانات: يتم استدعاء الدالة connect() من الكلاس Dbh لإنشاء اتصال بقاعدة البيانات.
  + استعلام SQL: يتم إعداد استعلام SQL باستخدام الدالة prepare() وتمريرها إلى الدالة execute() لتنفيذها. يتم استخدام عبارة SELECT لاسترداد بيانات المستخدمين من جدول المستخدمين.
  + التحقق من صحة كلمة المرور: يتم استخدام دالة password\_verify() للتحقق من صحة كلمة المرور المدخلة مع كلمة المرور المخزنة في قاعدة البيانات.
  + إعداد الجلسة: في حالة تطابق كلمة المرور، يتم إعداد متغيرات الجلسة لتخزين معلومات المستخدم المسجل الدخول.

<?php

class LoginModel extends Dbh{

protected function getUser($fullname,$password)

{

    $stmt=$this->connect()->prepare('SELECT \* FROM users WHERE user\_name=? OR email=?;');

    if(!$stmt->execute(array($fullname,$password))){

        $stmt=null;

        return true;

        exit();

    }

    if($stmt->rowCount()==0){

        $stmt=null;

        return true;

        exit();

    }

    $pwdhashed=$stmt->fetchAll(PDO::FETCH\_ASSOC);

    $checkpwd=password\_verify($password,$pwdhashed[0]["user\_password"]);

    if($checkpwd==false){

        $stmt=null;

        return true;

        exit();

    }elseif($checkpwd==true){

        $stmt=$this->connect()->prepare('SELECT \* FROM users WHERE user\_name=? OR email=? AND user\_password=?;');

        if(!$stmt->execute(array($fullname,$fullname,$pwdhashed[0]["user\_password"]))){

            $stmt=null;

            return true;

            exit();

        }

        if($stmt->rowCount()==0){

            $stmt=null;

            return true;

            exit();

        }

        $user=$stmt->fetchAll(PDO::FETCH\_ASSOC);

       session\_start();

        $\_SESSION['User\_ID']=$user[0]["id"];

        $\_SESSION['User\_Name']=$user[0]["user\_name"];

    }

    $stmt=null;

}

}