

## Sweetshelf - an online bookstore.

### The demonstration and description of its features and their development.

#### Table of contents:

*In order to fully access the website, please ensure you follow the instructions within the 'accessing the necessary files and data in order to run the site and test its functions' (page: 31) section from the table of contents.*

- **Languages & software used**
- **A brief summary of the features of the site.**
- **All features of the site:**
  - I. The database.
  - II. A list and description of all pages that make up the site:
    - sweetshelf.css
    - index.html
    - signup.html
    - userregister.php
    - login.php
    - emailredundant.html
    - landing page.html
    - Fiction page.html
    - Educational.html
    - kidsbooks.html
    - Non-fiction books.html
    - FAQ.html
    - About Us.html
    - Privacy Policy.html
    - Terms of Use.html
    - Genericbooklink.php
    - Purchase.php
    - Incorrectdetails.html
    - Thankyou.html
    - customer support.html
    - ticketsubmit.php
    - ticketredundant.html
    - ticketdetails.py
    - nosearchresults.html
- **Example demonstrations of the main features of the site (the tests for the features of the site).**
  - The registration of a new user.
    - Insertion into the database and continuation to login page.
  - The attempt to register a redundant email.

- The opportunity to return and re-attempt the registration with a different email.
  - The (valid) login of an existing user
    - Login and continuation to landing page.
    - Login and continuation to final purchase view.
  - The invalid login of an existing user
    - The opportunity to return and re-attempt the login from a separate viewed webpage.
  - The ability to view any displayed book as its own webpage, fetching data from the database's 'book stock' table.
  - The process of any (mock) purchase of a book from a user registered under North America.
    - £1.50 delivery fee.
  - The process of any (mock) purchase of a book from a user registered under UK.
    - No delivery fee. (£0.00)
  - The submission of a support ticket (from a fresh email).
    - Insertion of the ticket into the database and the ability to view its details on a separate webpage once it's been submitted.
  - The submission of a support ticket (from a redundant email).
    - **The opportunity to return and re-attempt the submission of the ticket from a separate viewed webpage.**
  - The successful search for a book.
    - The view of the book and the option to log in to purchase.
  - The unsuccessful search for a book
    - Viewing the notification of a lack of search results and suggestion to check spelling
  - The ability to log out (return to the original login page).
- **Accessing the necessary files and data in order to run the site and test its functions.**
    - Downloading the (royalty-free) book cover images and the webpage files and saving them to C:\xampp\htdocs
    - Downloading and importing the 'sweetshelf' database to localhost/phpmyadmin
  - **The development of the site**
    - I. Our group.
    - II. Our group meetings.

## Languages& Softwareused:

- **Languages**
  - HTML (creating the webpages).
  - CSS (look and formatting of the HTML).
  - Python (form handling).
  - PHP (form handling and SQL commands).
  - JavaScript (back-button function for pages following invalid form submission).
- **Software**
  - (Editing software - notepad++, sublimetext, etc.).
  - XAMPP (environment to test the website and MariaDB database server locally).
  - Apache (the local web server software).
  - phpMyAdmin (administration tool for managing the database server).
  - Microsoft Publisher (logo and other niche image designs).

## A brief summary of the featuresof the site:

- **A connected database** using three tables. These three tables contain (fabricated with royalty free cover art) book stock, registered user data and submitted support ticket data respectively. This database allows for SQL commands and data communication throughout the site.
- **A highly accessible and substantial set (21 visible, 24 total) of webpages** with a consistent visual layout.
- **The requirement to register (a non-redundant email) or enter a (valid – matching email and password against records of the ‘customers’ table of the database) log-in** prior to entering the site.
  - The data about the users is required to complete any purchase as well. Users are required to enter valid login details before finalizing any purchase.
  - If at any point, a registration fails due to a redundant email or a login fails because it is not identical to a record in the ‘customers’ table, the users have the option to immediately return and re-attempt their registration or login (by using a JavaScript function’s simulated back-button from the view of the ‘incorrect details’ webpage).

- **The ability to view any specific book as its own webpage** from any point of the site where they are displayed in isolation, with full detail on a separate webpage - fetching information from the book stock table using a SELECT SQL command on the condition a row corresponds to that book's name to obtain the attributes of the book. *This function's execution is followed by the ability to go through a purchase of the viewed book by logging In.*
- **The ability to make a mock purchase of any presented book (after viewing it in isolation).**
  - Once a user decides to view a book, they are prompted to login if they want to buy it.
  - After logging in, the user is directed to a separate webpage where the user's registered region is fetched using their email in a SELECT SQL command on the condition a row corresponds to that user's email to obtain their region.
  - On this webpage, the attributes corresponding to the name of the book are fetched similarly and a summary of the user's desired purchase is displayed – displaying the book's cover art, the book's name, its price and the delivery fee (obtained a condition set based off the user's region).
  - If the user then clicks 'Buy now!' (a submit-type input) on this same webpage below the information about their purchase, they are directed to the end of the mock purchase logic of this site – a page where the user is thanked for their purchase.
  - If at any point, a login fails because it is not identical to a record in the 'customers' table, the users have the option to immediately return and re-attempt their registration or login (by using a JavaScript function's simulated back-button from the view of the 'incorrect details' webpage).
- **The ability to submit and (view) tickets to 'customer support'.** Submitted tickets are stored (inserted) in the 'support\_tickets' table of the 'sweetshelf' database, and then fetched (selected) to be viewed.
  - The user enters their email, selects a ticket category by selecting from available radio button inputs, and then writes the content of their ticket into a large text-box. They click the (submit-input) button labelled as, 'Send'.
  - The user is directed to a separate webpage where the user's email, ticket content and ticket category are inserted into the 'support\_tickets' table as a single row (entry) with three values under three of the table's columns - 'email', 'ticket\_content' and 'ticket\_category' using an SQL insert command.
  - If the email of the user matches the value of an email of a row in the 'support\_tickets' table, the SQL insert command does not go through and instead of viewing the HTML of this webpage, the user views the

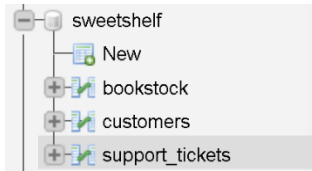
content of the 'ticket redundant' webpage where they have the option to immediately return and re-attempt their ticket submission.

- On this same webpage, the user is presented with a notification that their ticket has been submitted and then they are presented with the option to view their ticket by clicking a (submit-input) button labelled as, 'See your ticket details'.
  - The user is directed to a webpage where they can view the details of the successfully submitted ticket.
- 
- **The ability to search for any book available for sale using the search bar.**
    - On the majority of webpages, the user is presented with a search bar, giving them the ability to type in the title of any available book (non-case-sensitive) and be taken to a webpage for that book, where they can continue to purchase it.
    - If the user's search terms do not match a book title, they will view the 'nosearchresults.html' webpage, where they are told that their search got them no results and that they should mind their spelling. It also presents an additional search bar at the center of the webpage.
- 
- **The ability to log out (and be taken back to the original login page).**
    - On the majority of webpages, the user is presented with the option to click a 'log out' button and be taken back to the original login page.

***(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).***

All features of the site: (refer to **bold text** for each feature).

**A connected database named, 'sweetshelf'.** This database contains three tables:



- 'bookstock', which contains all the data about the books being sold on the bookstore, including their 'bookname', their 'author', their 'price', their 'category', their 'thumbnail' (which is the path to a saved image file corresponding to that book) as well as a unique 9-digit ID number to keep the entries unique.
- 'customers', which stores all data about users that register to the bookstore. This data includes their 'Email', their 'Name', their 'Password', their 'Date of Birth' and their 'Region'. As of submission, there is intentionally one entry in this table for demonstration purposes.

+ Options

	Email	Name	Password	Date of Birth	Region
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	test@gmail.com	test	test	2000-01-01	North America

- 'support\_tickets', which stores all data about tickets submitted by users, including their ticket's associated 'email', the submitted ticket's actual 'content', and the ticket's 'category' (which is inserted by a single radio button input from the four options: 'Delivery', 'Prices & Payment Methods', 'Returns & Pre-orders' and 'Security').

**The full description of each page is as follows:**

- 'sweetshelf.css' - the cascading style sheet that controls the presentation of the HTML in all pages. This contains all the attributes of the elements within the HTML.
- The index/front page - 'index.html'. This webpage contains a login form.
  - If the user logs in by inputting an email and a password that match the contents of a single record in the 'customers' table, they may proceed to the landing page.
  - The user is presented with a link they may click to be sent to the 'signup.html' page and 'sign up'.
  - If the details they submit for the login are incorrect, they view the contents of the 'incorrect details' webpage ('incorrectdetails.html')

where they are prompted to use a back-button to go back and re-attempt their login.

- The user's form input data is sent to the 'login.php' page by the 'post' method.
- 'signup.html' This webpage contains a registration form.
  - If the user registers by inputting a unique email followed by their details, they may proceed to the landing page ('landingpage.html').
  - The user is presented with a link they may click to be sent to the 'index.html' page and 'log in'.
  - The user's form input data is sent to the 'userregister.php' page.
  - If the email they submit for the sign up already matches against a row in the 'customer's' table, the SQL insert command does not work and the user views the contents of the 'emailredundant.html' webpage instead of the landing page. They are told their email is in use and prompted to use a back-button to go back and re-attempt their sign up.
- 'userregister.php' - the php page that handles the data from the registration form of 'signup.html' and then inserts it into the 'customers' table as a single row. If the email submitted for a registration is already registered, they view the content of the 'email redundant' webpage ('emailredundant.html') where they are told their email exists as a user and are prompted to use a back button to go back and re-attempt their registration.
- 'login.php' - the php page that handles the data from the login form of 'index.html' and then inserts it into the 'customers' table as a single row.
- 'emailredundant.html' - the webpage that the user is sent to if their entry into the registration form on 'index.html' is a duplicate entry – their email is redundant. This webpage features a back-button and a message explaining that the email has been registered, while encouraging the user to re-attempt their sign-up.
- 'landingpage.html' - the webpage that acts as the home page of the site. This is the page that a successful login or a successful registration leads a user to. It displays a few books from every genre ('category') that are stored in the 'bookstock' table of the database, as well as a 'new and trending' section which is exclusive to the home page. This webpage is the basis for the core structure of all webpages of the bookstore, the header and footer contain links to all immediately accessible webpages. The header contains a navbar, which contains its hyperlinks as well as the logo for 'Sweetshelf', which always acts

as a hyperlink for returning to this landing page (except for specific exceptions like 'incorrectdetails.html'). *Note: All webpages contain a similar header and footer to enable navigation throughout the site.*

- '[nosearchresults.html](#)' - The webpage designed to be viewed if the user's typed in search terms into the search bar cannot yield a book's title. It suggests they check their spelling and offers them with another search bar to re-attempt their search with.
- '[Fiction page.html](#)' - The genre webpage for all books with the 'category' value of 'Fiction'. This webpage displays all books under this category that are theoretically on offer for purchase and stored in the database. When the user wants to purchase any book, they click a form submit input button labelled 'Purchase', this form contains the hidden input of the book's name (as it is written in its row of the database), input under the name, 'bookname'. This form's action is the 'Genericbooklink.php' page (with the method, 'post'), when the user clicks the 'Purchase' button, the hidden input of the book's name is sent to 'Genericbooklink.php', where the data of the form can be handled and used (seen later).
- '[Educational.html](#)' - The genre webpage for all books with the 'category' value of 'Educational'. This webpage displays all books under this category that are theoretically on offer for purchase and stored in the database. When the user wants to purchase any book, they click a form submit input button labelled 'Purchase', this form contains the hidden input of the book's name (as it is written in its row of the database), input under the name, 'bookname'. This form's action is the 'Genericbooklink.php' page (with the method, 'post'), when the user clicks the 'Purchase' button, the hidden input of the book's name is sent to 'Genericbooklink.php', where the data of the form can be handled and used (seen later).
- '[kidsbooks.html](#)' - The genre webpage for all books with the 'category' value of 'Children's'. This webpage displays all books under this category that are theoretically on offer for purchase and stored in the database. When the user wants to purchase any book, they click a form submit input button labelled 'Purchase', this form contains the hidden input of the book's name (as it is written in its row of the database), input under the name, 'bookname'. This form's action is the 'Genericbooklink.php' page (with the method, 'post'), when the user clicks the 'Purchase' button, the hidden input of the book's name is sent to 'Genericbooklink.php', where the data of the form can be handled and used (seen later).



- 'Non-fiction books.html' - The genre webpage for all books with the 'category' value of 'Non-fiction'. This webpage displays all books under this category that are theoretically on offer for purchase and stored in the database. When the user wants to purchase any book, they click a form submit input button labelled 'Purchase', this form contains the hidden input of the book's name (as it is written in its row of the database), input under the name, 'bookname'. This form's action is the 'Genericbooklink.php' page (with the method, 'post'), when the user clicks the 'Purchase' button, the hidden input of the book's name is sent to 'Genericbooklink.php', where the data of the form can be handled and used (seen later).
- 'FAQ.html' - The webpage for Sweetshelf's frequently asked questions'. Displays generic questions and answers for the user.
- 'Terms of Use.html' - The webpage for Sweetshelf's terms and conditions. Displays generic terms and conditions for the user.
- 'About Us.html' - The webpage giving a brief insight into Sweetshelf's identity. Displays a generic background profile for the bookstore for the user.
- 'Privacy Policy.html' - The webpage for Sweetshelf's privacy policy. Displays a generic privacy policy for the user.
- 'Genericbooklink.php' - When any displayed book's 'Purchase' (submit input) button is clicked by a user, the form of that submit input's action is directed to this page. That same form sends the hidden input, 'bookname' which is identical to the desired book's stored name in the 'bookstock' table of the 'sweetshelf' database. This hidden input is handled (received via '\$\_POST [\_\_\_ ]') by 'Genericbooklink.php' and stored as a variable).
  - This variable is then used to execute SQL commands to fetch data about the desired book (I.E. thumbnail, author, price).
  - This data is then used in the printed HTML section of the 'Genericbooklink.php' webpage to display the relevant information about the book that the user has interest in buying – its title, its 'thumbnail' image, its author and its price.
  - The user then has another form presented to them – a login form, that they must fill if they wish to continue the purchase. This form also contains a hidden input of the specific book's name (named

'bookname'), equivalent to the previously stored variable (\$book\_name').

- The form requires them to input their 'email' and their 'password' (both inputs must match a record that exists in the 'customers' table to finish a purchase, but this is checked in Purchase.php).
  - After writing this information, the user can see a(submit input) button that they can push labelled as, 'Log In' that sends all inputs to 'Purchase.php' via the 'post' method.
  - The form's action is set to direct the user to 'Purchase.php' once this is done. 'Purchase.php' receives 'bookname' (the name of the book), 'email' (the user's email) and 'password' (the user's password) and then handles the data for its purposes (seen later).
- 'Purchase.php': Any instance of 'Genericbooklink.php' contains a login form that sends three items of data via the post method to this page. These three items of data are, 'bookname' (the name of the book the user is interested in buying), 'email' (the email of the user) and 'password' (the password of the user). These items of data are all stored as variables (the 'email' is stored as \$email, the 'bookname' is stored as \$book\_name and the received password is stored as \$inputpassword').
    - The email variable, '\$email' is then used in a SQL command to select results from the 'Password' column where the corresponding email is equivalent to '\$email' and the fetched row's 'Password' is then stored as, '\$truepassword'. In order to echo (print) any HTML on this page, '\$truepassword' must be equivalent to '\$inputpassword', otherwise the user views the 'incorrectdetails.html' page and cannot view 'Purchase.php' at all.
    - The email variable, '\$email' is then used in another SQL command to select the user's 'Region' from the 'customers' table (it selects the region from the row where the 'Email' column is equal to '\$email'). The fetched value from this SQL command – the user's region – is then stored as, '\$region'.
    - The variable, '\$region' is then used to set a value for the delivery fee (stored as '\$delivery') the user would have to pay. '\$region' acts as the deciding factor between 6 "if" conditions, in which the delivery fee is set depending on the value of the user's region.
    - The input 'bookname' stored variable, '\$book\_name' is used in an SQL command to select the results from the 'thumbnail' and 'price' columns where the 'bookname' column is equal to the '\$book\_name' variable.
    - The fetched row's values under the columns, 'thumbnail' and 'price' (which is stored as '\$book\_price') are used in the displayed HTML of the webpage (along with the prior mentioned '\$delivery' and '\$book\_name' variables) to display a summary of the user's purchase.
    - The user is presented with one final (input submit) button labelled, "Buy now!" The form possessing this input has the form action,

'Thankyou.html' using the method, 'get' so that upon clicking 'Buy now!' to finalize the purchase.

- The user is directed to 'Thankyou.html' (seen later).
- 'incorrectdetails.html' - throughout various areas of the website, it is necessary to provide the user with a notification of "invalid details" being entered into a form submission, as well as the opportunity to reattempt that form submission.
  - This page provides the notification, "The details you have entered are invalid. Please return and try again."
  - This page contains a JavaScript function that allows for a displayed button directly below the notification. This button allows the user to return to the previous URL in the user's browser history upon being clicked.
  - The hyperlinks are intentionally altered to reference "#", to maintain the integrity of the site.
- 'Thankyou.html' - The end of any mock purchase within this site. Upon confirming the purchase, the user is directed here from 'Purchase.php' and greeted by a statement of gratitude, 'Thank you for your purchase.' at the center of the webpage. *Note: like all pages (with few exceptions), the header and footer keep the rest of the site fully accessible, the user may continue how they desire from this point.*
- 'customersupport.html' - The webpage that provides users the opportunity to submit support tickets (to the 'ticket\_support' table in the 'sweetshelf' database). *This webpage is accessed via a hyperlink in the headers of this site.*
  - This webpage presents users with a form where the user submits their ticket by entering an 'email' value, entering a 'ticket\_category' value by selecting from a range of four radio buttons (correlating to an input of, 'Delivery', 'Prices & Payment Methods', 'Returns & Pre-orders' or 'Security'), typing something related to their issue into the 'ticket\_content' input text box, and then clicking the 'Send' (submit input) button.
  - Below this form is the notification: 'Please be aware that you can only submit one valid ticket at once!') If the user does input a redundant email address in their submission of the form, they view 'ticketredundant.html' within 'ticketsubmit.php', instead of displaying the HTML of 'ticketsubmit.php'.
  - The action of the form is 'ticketsubmit.php' via the method, 'post'. The prior quoted values: 'email', 'ticket\_category', 'ticket\_content' are sent to 'ticketsubmit.php' (usage of this data is seen later).

- 'ticketsubmit.php' - This webpage is accessed following a submission of a ticket from the form on 'customer support.html'. 'customer support.html' sends this webpage three items of data; named, 'email', 'ticket\_category' and 'ticket\_content' respectively (the user's submitted email, the category of their ticket and the content of the ticket). These are stored as the variables, '\$email', '\$ticket\_content' and '\$ticket\_category'.
  - These variables are then used to execute an SQL command where they are inserted as a single row into the 'support\_tickets' table of the 'sweetshelf' database under the columns, 'email', 'ticket\_content' and 'ticket\_category' respectively.
  - If this SQL command is successful (there are no problems with a redundant email), the webpage continues, and the HTML of 'ticketsubmit.php' is displayed to the user. Otherwise, the user viewsthe content of 'ticketredundant.html'.
  - The HTML of 'ticketsubmit.php' contains a notification that their ticket has been submitted - 'Your ticket has been submitted'.
  - The user is also presented with a (submit input) button labelled, 'See your ticket details', the form of which contains hidden inputs of the ticket's content, the ticket's category, and the user's email (\$ticket\_content, \$ticket\_category and \$email). The action of this form is 'ticketdetails.py' via the method, 'post'.
  - If the user chooses to click this button, they are directed to 'ticketdetails.py' where the data of the form is handled and used to display the details of their ticket.
- 'ticketredundant.html'- If the user inputs a redundant 'email' value into the 'support\_tickets' table (the SQL command on 'ticketsubmit.php' fails), the user is directed to this webpage.
  - This page provides the notification, "You have already submitted a ticket! Please wait until one of our customer support team addresses your issue."
  - This page contains a JavaScript function that allows for a displayed button directly below the notification. This button allows the user to return to the previous URL in the user's browser history upon being clicked.
  - The hyperlinks are intentionally altered to reference "#", to maintain the integrity of the site.
- 'ticketdetails.py' - Any instance of 'ticketsubmit.php' sends three values to this webpage, named: 'email', 'ticket\_content' and 'ticket\_category'.
  - These values correlate to a single ticket submitted by a user – their email, the content of the ticket and the category of the ticket.

- This webpage stores these as the variables, 'email', 'ticket\_category' and 'ticket\_content'.
- This webpage's displayed HTML contains the details of their submitted ticket using these variables and the user can see the ticket they submitted to Sweetshelf's database. *Note: like all pages (with few exceptions), the header and footer keep the rest of the site fully accessible, the user may continue how they desire from this point.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

A demonstration of the main features of the site (the tests for the features of the site):

### Examples of the main features of the site in action (tests).

- The registration of a new user.
  - Insertion into the database and continuation to landing page.

<ul style="list-style-type: none"> <li>sweetshelf           <ul style="list-style-type: none"> <li>New</li> <li>bookstock</li> <li>customers</li> <li>support_tickets</li> </ul> </li> <li>test</li> </ul>	+ Options				
	<div> <div> <div></div> <div></div> <div></div> </div> <div>Email</div> <div>Name</div> <div>Password</div> <div>Date of Birth</div> <div>Region</div> </div>				
	<div> <div> <div></div> <div></div> <div></div> </div> <div>testna@gmail.com</div> <div>test</div> <div>test</div> <div>2000-01-01</div> <div>North America</div> </div>				
	<div> <div> <div></div> <div></div> <div></div> </div> <div>testredundant@gmail.com</div> <div>testredundant</div> <div>testredundant</div> <div>2000-01-01</div> <div>UK</div> </div>				

*'Customers' table before registration*

SWEETSHelf

SIGN UP:

Please enter your email:

Please enter your name:

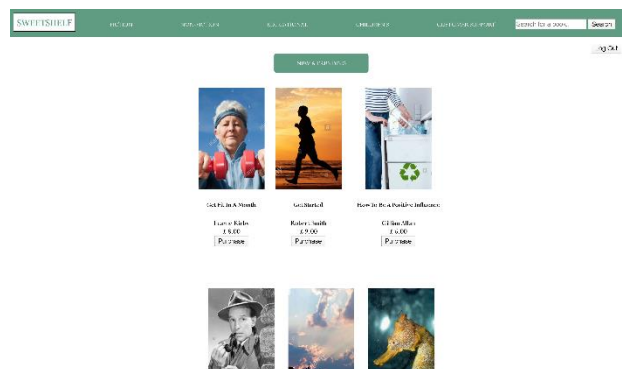
Please enter a password:

What is your date of birth?:

Which region do you live in  

Europe ☐
UK ☒
Asia ☐
Africa ☐
North America ☐
South America ☐

The new user's entered data before clicking 'Sign Up!' and submitting their form data.



*The landing page, where the user is directed to on a successful registration*

sweetshelf

New

bookstock

customers

support\_tickets

test

university

+ Options

←

→

▼

Email

Name

Password

Date of Birth

Region

☐

Edit

Copy

Delete

newuser@gmail.com

newuser

newuser

2000-01-01

UK

☐

Edit

Copy

Delete

testna@gmail.com

test

test

2000-01-01

North America

☐

Edit

Copy

Delete

testredundant@gmail.com

testredundant

testredundant

2000-01-01

UK

☐

Edit

Copy

Delete

testuk@gmail.com

testuk

testuk

2000-01-01

UK

*The new user's registered data stored in the 'customers' table of the database.*

- **The invalid registration of a redundant email.**
  - **Failure of the SQL command and the opportunity to return and re-attempt the registration with a different email.**

sweetshelf

New

bookstock

customers

support\_tickets

+ Options

Email

Name

Password

Date of Birth

Region

Edit

Copy

Delete

testna@gmail.com

test

test

2000-01-01

North America

Edit

Copy

Delete

testredundant@gmail.com

testredundant

testredundant

2000-01-01

UK

*'testredundant@gmail.com' entry's existence prior to registration.*



SIGN UP:

Please enter your email:

testredundant@gmail.com

Please enter your name:

testredundant

Please enter a password:

testredundant

What is your date of birth?

01/01/2000

Which region do you live in?

- Europe ☐
- UK ☒
- Asia ☐
- Africa ☐
- North America ☐
- South America ☐

Sign Up!

*The redundant user's duplicate email value and the rest of their data before clicking 'Sign Up!' and submitting their form data.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

THAT EMAIL IS IN USE. PLEASE REGISTER WITH A DIFFERENT EMAIL ADDRESS.

Return

*The display after submitting the duplicate entry.*

- The (valid) login of an existing user
  - Login and continuation to landing page.



name	email	password	date of birth	region
newuser	newuser@gmail.com	newuser	2000-01-01	UK

*The registered, 'newuser@gmail.com' row in the 'customers' table.*



WELCOMETO SWEETSHelf!

LOG IN:

Enter your details below:

Please enter your email:

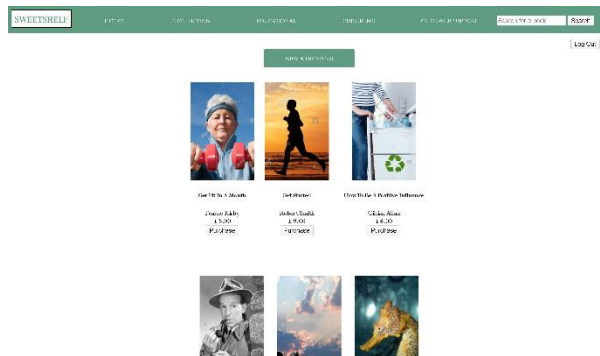
newuser@gmail.com

Please enter your password:

newuser

Log In




*The user's correct input details prior to clicking the 'log in' submit button.*



*The landing page after the user logged in properly.*



○ **Login and continuation to final purchase view.**

<input type="checkbox"/>	 Edit	 Copy	 Delete	testuk@gmail.com	testuk	testuk	2000-01-01	UK
--------------------------	--	--	--	------------------	--------	--------	------------	----

*The 'testuk@gmail.com' user's entered existing details in the 'customers' table.*

CURIOUS SEAHORSE



BY: HANNO  
ROSEN

£5.00

LOG IN & CONTINUE

Please enter your email:

testuk@gmail.com

Please enter your password:


testuk

Log In

*'testuk@gmail.com' user's details in the login form prior to initiating a purchase by pushing the 'log in' submit button.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

YOUR PURCHASE



Curious Seahorse

Price: £5




Delivery Fee: £0

Buy now!

(Please be aware that the delivery fee may vary based on your region)

*‘The following purchase view after the successful login’.*

- The invalid login (wrong email and/or wrong password) of an existing user
  - A lack of access to the landing page or the next stage of a purchase and the opportunity to return and re-attempt the login from a separate viewed webpage.

<input type="checkbox"/>	 Edit	 Copy	 Delete	testuk@gmail.com	testuk	testuk	2000-01-01	UK
--------------------------	--	--	--	------------------	--------	--------	------------	----

*The ‘testuk@gmail.com’ user’s entered existing details in the ‘customers’ table.*

WELCOME TO SWEETSHIELD!

LOG IN:

Enter your details below:

Please enter your email:

testuk@gmail.com

Please enter your password:

incorrectpassword

Log In

Or Sign Up!

WELCOME TO SWEETSHIELD!

LOG IN:

Enter your details below:

Please enter your email:

invalidemail@gmail.com

Please enter your password:

testuk

Log In

Or Sign Up!

WELCOME TO SWEETSHIELD!

LOG IN:

Enter your details below:

Please enter your email:

invalidemail@gmail.com

Please enter your password:

incorrectpassword

Log In

Or Sign Up!

*incorrect password*

*incorrect email  
password*

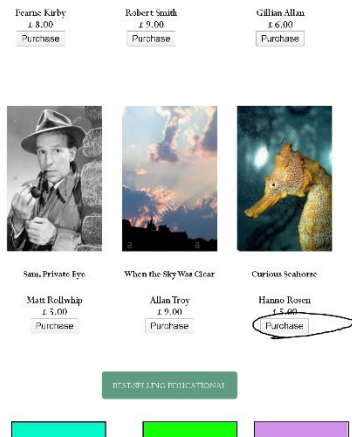
*incorrect email and*

THE DETAILS YOU HAVE ENTERED ARE INVALID. PLEASE RETURN AND TRY AGAIN.

Return

*all entries into login forms that do not match a record perfectly result in the above display. The user has the option to return and re-attempt their login.*

- The ability to view any displayed book as its own webpage, fetching data from the database's 'book stock' table.



*To be directed to a PHP webpage that fetches a book's data and displays it in isolation, the user must click the 'Purchase' submit button. (The form possessing this 'purchase' submit button also sends a hidden input of the book's name, as it is stored in the database server). As seen below:*

```
<input type = "hidden" value = "Curious Seahorse" name = "bookname"> <input type = "submit" value = "Purchase" name = "display">
```

100000065 Curious Seahorse Hanno Rosen 5 Children's seahorse.jpg

*The entry in the 'bookstock' table associated with this book.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

CURIOUS SEAHORSE



BY: HANNO  
ROSEN

£5.00

LOG IN & CONTINUE

Please enter your email:

Please enter your password:

Log In

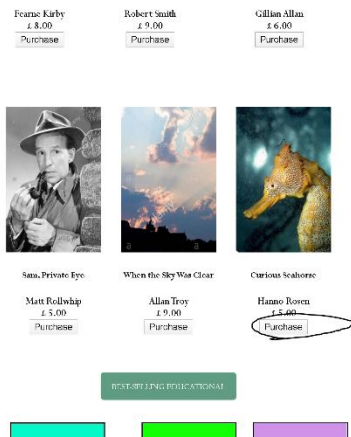
*The book's data is obtained from the database and displayed to the user using a SQL SELECT command within this PHP page based off of the data sent to the page from the prior form (with the hidden input of the book's name). As seen below:*

```
//defining the SQL request to obtain the values under the columns: 'thumbnail, price and author' where  
//The SQL request is set as the variable, '$sql'  
$sql = "SELECT thumbnail, price, author FROM bookstock WHERE bookname = ' " . $book_name . "'";  
  
if(isset($_POST['bookname'])) && $_POST['bookname'] != '' && $_POST['bookname'] != null) {  
    $book_name = $_POST['bookname'];  
    /* '$book_name' variable = the value sent by post method under the name 'bookname'. This is the name of the book the user clicked on.}
```

- The process of any(mock)purchase of a book from a user registered under North America.
  - £1.50 delivery fee.

<input type="checkbox"/>	Edit	Copy	Delete	testna@gmail.com	test	test	2000-01-01	North America
--------------------------	------	------	--------	------------------	------	------	------------	---------------

*The user registered with the email, 'testna@gmail.com' and their details. They are from North America.*



*To initiate the process of buying a book, the user must first view it in isolation on a separate webpage by clicking 'purchase'.*



*They are then prompted to login if they want to continue with a purchase.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

Please enter your email:

testna@gmail.com

Please enter your password:

test

Log In

*The user types in their details and clicks 'log in' to submit the form.*

YOUR PURCHASE



Curious Seahorse

Price: £5

Delivery Fee: £1.5

Buy now!

(Please be aware that the delivery fee may vary based on your region)

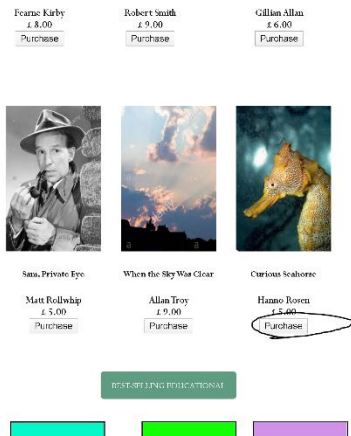
*The user is then presented with the final purchase view before finishing it by clicking 'Buy now!' Their delivery fee is set based off their region. Because this person is from North America, their delivery fee is £1.5. As seen in the below condition:*

```
else if ($region == "North America") {  
    //if region of the user is stored as North America, the delivery fee will be 1.5.  
    $delivery = 1.5;  
}
```

- The process of any (mock) purchase of a book from a user registered under UK.
  - No delivery fee. (£0.00)

□ Edit Copy Delete testuk@gmail.com testuk testuk 2000-01-01 UK

The user registered with the email, 'testuk@gmail.com' and their details. They are from the UK.



To initiate the process of buying a book, the user must first view it in isolation on a separate webpage by clicking 'purchase'.



They are then prompted to login if they want to continue with a purchase.

CURIOUS SEAHORSE



BY: HANNO  
ROSEN

£5.00

LOG IN & CONTINUE

Please enter your email:

testuk@gmail.com

Please enter your password:

testuk

Log In

*The user types in their details and clicks 'log in' to submit the form.*

**(NEXT SECTION BELOW ON NEW PAGE, KEEP  
SCROLLING).**





(Please be aware that the delivery fee may vary based on your region)

The user is then presented with the final purchase view before finishing it by clicking 'Buy now!' Their delivery fee is set based off their region. Because this person is from the UK, their delivery fee is £0. As seen in the below condition:

```
$delivery = 1.5;  
} else if ($region == "UK") {  
    //if region of the user is stored as UK, the delivery fee will be 0.00.  
    $delivery = 0.00;  
}
```

- The submission of a support ticket (from a fresh email).
  - Insertion of the ticket into the database and the ability to view its details on a separate webpage once it's been submitted.

	email	ticket_content	ticket_category
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	testredundant@gmail.com	testredundant	Prices & Payment Methods

The 'support\_tickets' table prior to the submission of this new support ticket

HOW CAN WE HELP?

Here at Sweetshelf, we make sure our customers leave our stores (both online and offline) completely satisfied. We understand that sometimes you can run into problems with the products and services we want to supply you with, please don't be too shy to send in a support ticket for help.

SEND IN A TICKET & WE'LL GET BACK TO YOU!

Please enter your email:

newticket@gmail.com

Where are you having trouble?

Delivery

Prices & Payment Methods

Returns & Pre-orders

Security

Tell us what's wrong:

New ticket

Send

(Please be aware that you can only submit one valid ticket at once!)

The user's entered ticket prior to submitting it by pushing the submit 'send' button.

YOUR TICKET HAS BEEN SUBMITTED;

See your ticket details

The '.php' webpage the user is directed to after submitting their valid form entry that handles their form data and then executes an insert SQL command. The user has the option to be directed to a separate '.py' webpage where they can review their submitted ticket's details.

		email	ticket_content	ticket_category
<input type="checkbox"/>	Edit  Copy  Delete	newticket@gmail.com	New ticket	Delivery
<input type="checkbox"/>	Edit  Copy  Delete	testredundant@gmail.com	testredundant	Prices & Payment Methods

(Their entry has been inserted into the 'support\_tickets' table)

## YOUR SUPPORT TICKET DETAILS:

Your email: newticket@gmail.com

Category of ticket: Delivery

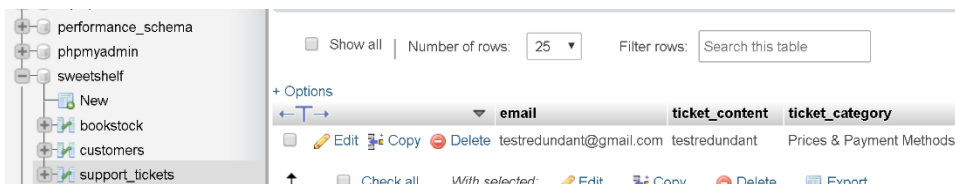
Description of ticket:

'New'

*the view of the user's submitted ticket after clicking 'see your ticket details' (based off the form data submitted in the 'see your ticket details' button's form's hidden inputs). As seen below:*

```
echo "<FORM action='ticketdetails.py' method='POST'>";
echo "<input type = 'hidden' value = \" , $email, \" name='email'> <br>";
//enters the variable '$email' into the hidden input, '$email' comes from the 'email' value the user sends
//Set via '$email = $_POST['email'];'
echo "<input type = 'hidden' value = \" , $ticket_category, \" name='ticket_category'>";
//enters the variable '$ticket_category' into the hidden input, '$ticket_category' comes from the 'ticket_category' value the user sends
//Set via '$ticket_category = $_POST['ticket_category'];'
echo "<input type = 'hidden' value = \" , $ticket_content, \" name='ticket_content'>";
//enters the variable '$ticket_content' into the hidden input, '$ticket_content' comes from the 'ticket_content' value the user sends
//Set via '$ticket_content = $_POST['ticket_content'];'
echo "<p class = 'center'> <input type='submit' value='See your ticket details'> </p>";
//by clicking the 'See your ticket details' submit input, the user is taken to the 'ticketdetails.py' file
```

- The invalid submission of a support ticket (from a redundant email).
  - Failure of insert command and the opportunity to return and re-attempt the submission of the ticket from a separate viewed webpage.



email	ticket_content	ticket_category
testredundant@gmail.com	testredundant	Prices & Payment Methods

*The existing 'ticketredundant@gmail' entry in the 'support\_tickets' table prior to the attempt to insert the duplicate entry with the redundant email.*

#### HOW CAN WE HELP?

Here at Sweetshelf, we make sure our customers leave our stores (both online and offline) completely satisfied. We understand that sometimes you can run into problems with the products and services we want to supply you with, please don't be too shy to send in a support ticket for help.

#### SEND IN A TICKET & WE'LL GET BACK TO YOU!

Please enter your email:

testredundant@gmail.com

Where are you having trouble?

Delivery

Prices & Payment Methods

Returns & Pre-orders

Security

Tell us what's wrong:

testredundant

Send

(Please be aware that you can only submit one valid ticket at once!)

*The 'testredundant@gmail.com' duplicate entry prior to submitting by pressing 'send'.*

YOU HAVE ALREADY SUBMITTED A TICKET! PLEASE WAIT UNTIL ONE OF OUR CUSTOMER SUPPORT TEAM ADDRESSES YOUR ISSUE.


Return

*any duplicate email entry into the 'support\_tickets' table fails and this webpage is displayed instead of the webpage confirming the user's ticket submission and offering an option to then view it. The user has the option to return and re-attempt their ticket submission.*


- The ability to search for any book available for sale using the search bar

[NON-FICTION](#) [FICTIONAL](#) [CHILDREN'S](#) [CUSTOMER SUPPORT](#)


[NEW & TRENDING](#)



Get Fit In A Month  
Fennee Kirby  
£ 8.00

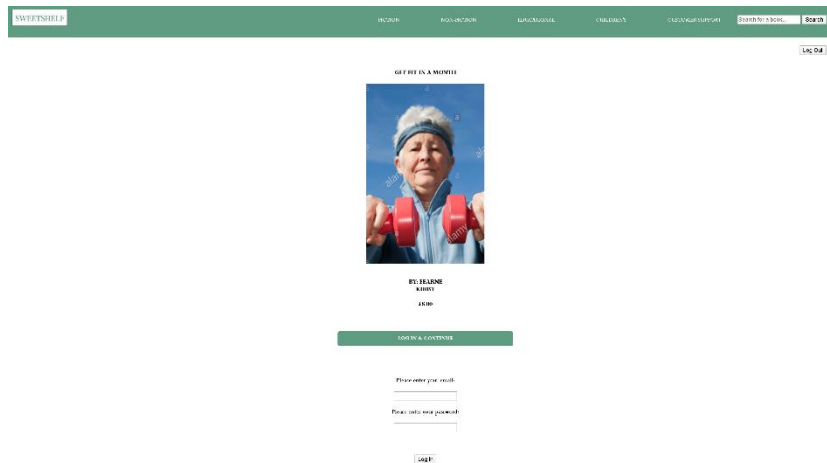


Get Started  
Robert Smith  
£ 9.00



How To Be A Positive Influence  
Gillian Allan  
£ 6.00

*The search for, 'Get Fit In A Month' before clicking the 'search' submit button.*



*The result for, 'Get Fit In A Month' after clicking the 'search' submit button.*

- The result of search terms that do not match a book title.



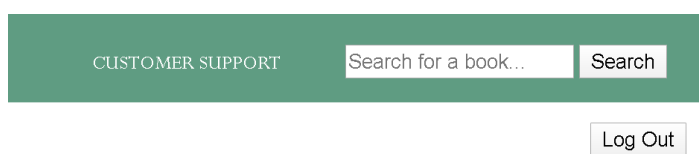
*The search for the non-existing book, 'banana'.*

SORRY, WE COULDN'T FIND ANY CONTENT FOR YOUR SEARCH.

(Remember to check your spelling)

*The result of search terms that do not match a book title.*

- The ability to log out (and be taken back to the original login page).



*The 'Log Out' (submit-input) button before being clicked.*

WELCOME

SWEETSHelf

WELCOME TO SWEETSHelf!

LOG IN:

Enter your details below:

Please enter your email:

Please enter your password:

Log In

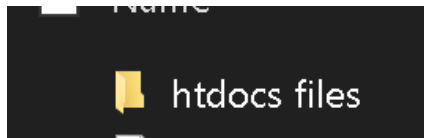
Or Sign Up!

*The result of clicking the, 'log out' button. The user is directed back to the original login page.*

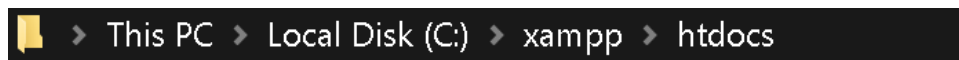
**(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).**

## Accessing the necessary files and data in order to run the site and test its functions:

### Downloading the (royalty-free) book cover images and the webpage files and saving them to C:\xampp\htdocs



- Within the submitted ZIP archive, there is a file named, 'htdocsfiles'. This file contains exactly all files that need to be saved to C:\xampp\htdocs.
  - (This includes the image '.jpg' files corresponding to the book stock stored in the 'bookstock' table of the database).
  - (This also includes all files that make up the webpages – under the extensions: '.html', '.php' and '.py').
- Copy **all** these files to the following (or functionally identical) path:



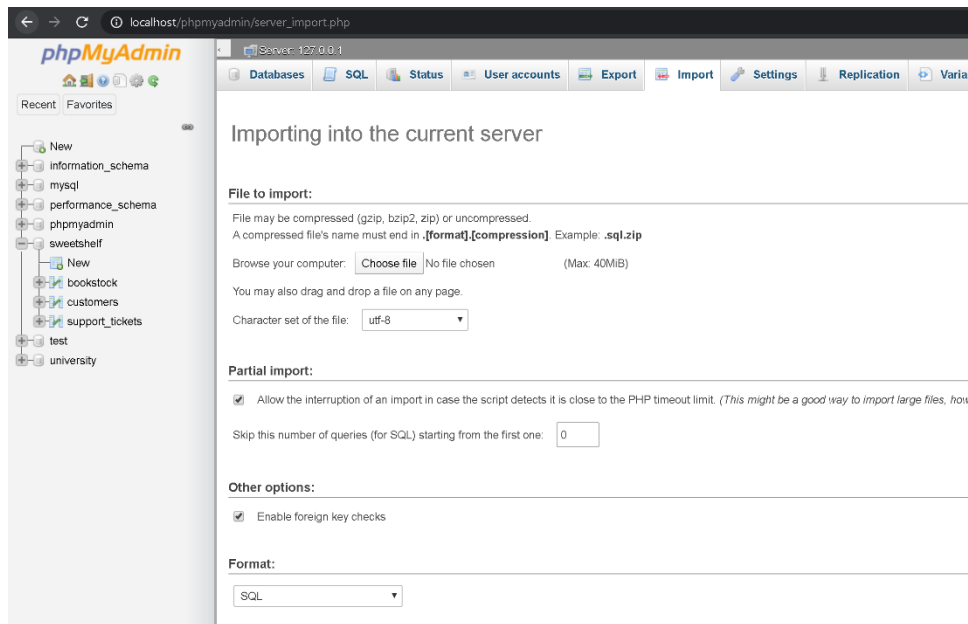
*'This PC > Local Disk (C:)>xampp>htdocs'*

### Downloading and importing the 'sweetshelf' database to localhost/phpmyadmin

- locate the 'sweetshelf.sql' SQL file within the submitted ZIP archive.



- Ensure it is saved somewhere and then access the 'phpMyAdmin' administration tool for MySQL databases (manageable at localhost/phpmyadmin), by typing the address into your (compatible) browser or clicking 'Admin' in the 'MySQL' module's row of the XAMPP control panel.
- Once on the administration tool, click 'New' over the list of present databases and then click 'import'.



- From the import menu, next to 'Browse your computer', there is a 'Choose file' button. Click it, and then select the SQL file from where it was saved.
- Ensure the 'format' option, 'SQL' is selected.
- Go to the bottom of the menu and click 'Go'.
- A new database entry will appear in the administration tool named, 'sweetshelf' and thus the database has been successfully obtained for the purposes of running and testing this site.

***(NEXT SECTION BELOW ON NEW PAGE, KEEP SCROLLING).***



## The development of the site:

**Our group ('Team Sweetshelf'): Mohammed A. Elkkari (Kaplan ID: P165798), Katrina Nsomere (Kaplan ID: P266084) & Meshaal Albaker (Kaplan ID: P168606)**

### Shared contributions (during group meetings and communications):

- Decisions on the additional pages to add (e.g. 'FAQ', 'Terms & Conditions')
- The general style and layout of the webpages of the website.
- The map and relationships between all webpages of the website.
- The design of the tables of the database.
- Logical (abstract) designs of the features.
  - How a purchase would work on the site.
  - How a registration would work on the site.
  - How a log-in would work on the site.
  - How a ticket submission would work on the site.

### Katrina Nsomere (Kaplan ID: P266084)

#### **Individual Contributions: (\* = group-related)**

- Designing the 'Sweetshelf' logo, and the 'Welcome' banner.
- Writing the CSS for the general use 'center' class and 'stretch-' classes (used to adjust margins at certain instances).
- Writing CSS classes for <h3> and <h4> tags.
- Writing the CSS classes for the division tags of our footers (labelled in-code as variants of 'bottom' classes)
- Writing the HTML of the **landing page**.
- Writing the HTML of the 'ticket redundant', 'email redundant' and 'incorrect details' webpages using (\*M.A.Elkkari's JavaScript function to enable) a back-button on these pages.
- Writing the HTML for the '**About Us**' page.
- Writing the HTML for the '**Terms & Conditions**' page.

### Meshaal Albaker (Kaplan ID: P168606)

#### **Individual Contributions: (\* = group-related)**

- Deciding on generic book names, corresponding categories for these names, author names and prices to obtain around 78 books. \*Then, sharing these to be inserted into the database and integrated into our website.

- Obtaining royalty free image files related to the generic book names and \*sharing their saved path to be inserted into the 'bookstock' table to allow for the fetching of images on the website.
- Finding good reference websites and writing content for our additional pages - 'FAQ', 'Terms & Conditions' and 'Privacy Policy'.
- Writing the HTML for the **'FAQ'** page.
- Writing the HTML for the **'Privacy Policy'** page.
- Writing the HTML for the **'Thank you'** page.

Mohammed A. Elkkari (Kaplan ID: P165798)

**Individual Contributions:** (\* = group-related)

- Python-enabled and PHP-enabled webpages.
  - **'userregister.php'**,
  - **'Login.php'**,
  - **'Genericbooklink.php'**,
  - **'Purchase.php'**,
  - **'ticketsubmit.php'**,
  - **'ticketdetails.py'**
    - Connection to the database server and SQL commands in PHP; inserting and selecting data into and from the database server (during any live instance of the website).
    - Form-handling in webpages using PHP or Python.
    - Writing the printed HTML of these Python-enabled/PHP-enabled webpages.
- Introducing a method of allowing a user to re-attempt any invalid form submissions
  - (Ended up being using JavaScript to make a back-button function).
- Writing the HTML of the books' 'genre' webpages:
  - **Fiction** (fiction page.html)
  - **Non-Fiction** (Non-fiction books.html)
  - **Educational** (Educational.html)
  - **Children's** (kidsbooks.html)
- Writing CSS of general element traits, (main classes intended for the <div> tag) 'wrappers' for specific elements (e.g. 'book image) and traits of the headers and nav-bars of webpages.
  - +CSS classes for specific niche elements like the classes: 'logo', 'ticketcomment', etc. *Note: this continues to the next page*
- Writing the HTML of the **'customer support.html'** webpage.
- Writing the printed HTML of the **'Purchase.php'** page.
- Writing the printed HTML of the **'Genericbooklink.php'** page.
- Writing the printed HTML of the **"ticketsubmit.php"** page.
- Writing the printed HTML of the **"ticketdetails.py"** page.
- Writing the HTML of the **"index.html"** page.

*(Mohammed A. Elkkari – Kaplan ID: P165798)*

- Writing the HTML of the “**signup.html**” webpage.
- Writing the HTML of the search bar that is present on most webpages.
- Writing the HTML of the log-out button that is present on most webpages.
- Writing the HTML of the “**nosearchresults.html**” webpage.
- Running and testing the site’s functions throughout the project (possessing the relevant data in the connected database server and a reliable XAMPP installation).
- Fixing and finalizing HTML, CSS, PHP and Python while running the website to find errors.
- \*Writing comprehensive notes over code.
- \*Writing documentation, displaying demonstration and outlining replication instructions for the purpose of the project’s submission and presentation.
- \*Scheduling and recording meetings.

### **Our group meetings:**

- 15/03/2020 -
  - Location/platform: The library on campus.
  - Attendance:
    - Mohammed A. Elkkari (Kaplan ID: P165798)
    - Katrina Nsomere (Kaplan ID: P266084)
    - MeshaalAlbaker (Kaplan ID: P168606)
  - Decisions& Tasks:
    - The map and relationships between all webpages of the website.
    - The general style and layout of the webpages of the website.
- 17/03/2020 -
  - Location/platform: The library on campus.
  - Attendance:
    - Mohammed A. Elkkari (Kaplan ID: P165798)

- Katrina Nsomere (Kaplan ID: P266084)
  - MeshaalAlbaker (Kaplan ID: P168606)
- Decisions& Tasks:
  - The design of the tables of the database ('customers' and 'bookstock').
  - How a registration would work on the site.
  - How a log-in would work on the site.
- 28/03/2020 -
  - Location/platform: Zoom
  - Attendance:
    - Mohammed A. Elkkari (Kaplan ID: P165798)
    - Katrina Nsomere (Kaplan ID: P266084)
    - MeshaalAlbaker (Kaplan ID: P168606)
  - Decisions& Tasks:
    - How a purchase would work on the site.
    - Decisions on the additional pages to add (e.g. 'FAQ', 'Terms & Conditions')
- 29/03/2020/
  - Location/platform: Zoom
  - Attendance:
    - Mohammed A. Elkkari (Kaplan ID: P165798)
    - Katrina Nsomere (Kaplan ID: P266084)
    - MeshaalAlbaker (Kaplan ID: P168606)
  - Decisions& Tasks:
    - How a ticket submission would work on the site.
    - The table design for submitted tickets' table, 'support\_tickets'.