

*Student Name:*

Robert Vaughan

*Student Number:*

C15341261

*Title:*

Web Development II Assignment Report

***Contents***

Design……………….………………………………………………………………………………………...….3

Source Explanation.…..……………………………………………………………………...…………....5

User-End Experience…………………………………………………………………………………...….6

List of Source Code.……………………………………………………………………………………....13

**Design**

When creating the website, I was aware that for the site to be fully functional, it must appeal to the aesthetic of the user. A user must be able to navigate through the site without strain and to be able to make requests with ease. However, an expert user/designer must be aware of what is required of the backend and how to make it simple for it to function properly. A guide for both front and back end design is Nielsen’s Heuristics. Below is an analysis of each Heuristic and how it relates to the following assignment.

1. **Visibility of System Status**

The user must be well informed of all actives within the backend of the site (if there are any) and must be informed of any activity clearly. This can be implemented when a user is searching for a book. The site during this process must inform the user if it was successful in finding a list of books that relate to their search query.

1. **Match between system and the real world**

If the user is presented with a message sent by the server, the message should be clear to the user and in generalised terms to which they can understand.

1. **User control and freedom**

A user has the option to leave at any point and is not constrained to an aspect of the site, this can be provided by offering a navigation bar to the user that has different page links and an option to log out.

1. **Consistency and standards**

The user should be well informed that different actions create a different result. This shall be applied to the register page for a user must be informed each error they made in the data they submitted to the page. If one a generalised error message was presented, a user would not know where there error lies.

1. **Error prevention**

Error prevention is necessary for a user should not be able to store in accurate or incorrect data into the database. Therefore, the backend must check the user’s inputs to ensure they are valid and to display appropriate error messages if this occurs.

1. **Recognition rather than recall**

The user’s memory must not be strained while navigating through the site. The user should not be forced to learn aspects of the site in order to function it. The site should keep a simplistic approach.

1. **Flexibility and efficiency of use**

The site should accessible to users of all skill level. This again enforces the approach of keeping the site simple and not complex.

1. **Aesthetic and minimalist design**

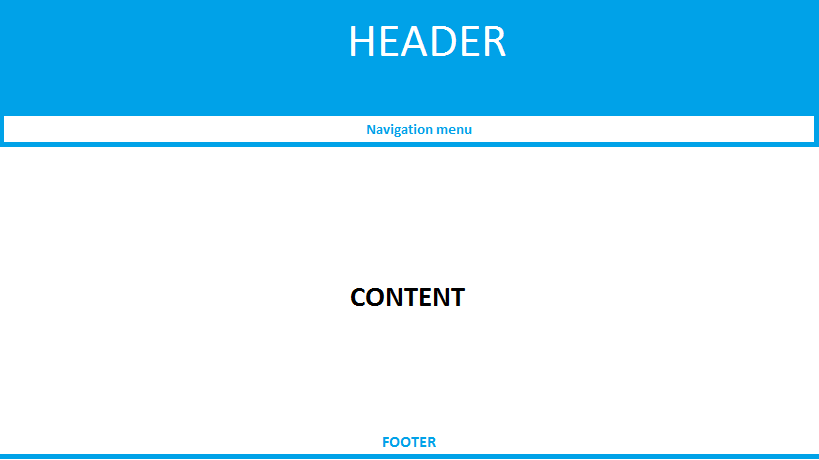
This heuristic is very important in relation to the library site for if a designer uses an over complicated system and/or design, users and expert users will find it difficult to make use of the sites functionalities.

1. **Help users recognize, diagnose, and recover from errors**

The user should not be presented with complex error messages that are only useful to an expert user. All error messages, such as a failed SQL query to the database, must be outputted to the user and explain what has occurred on a basic level.

1. **Help and documentation**

A site should provide a user with basic guides to help him or her understand the site and its functionality if required. This can be achieved with the user of well documented code. In terms of publicised documentation, the site should be relatively simplistic so that not guide is required.



I believe that the warm light welcoming colours and the contrast of the light blue to white suited is a good combination. The use of black and perhaps red for errors would also naturally inform the user that information highlighted in those colours was critical and/or important. Therefore, all content and information will be displayed in these colours. A valid implementation would be to use the <div> element to control the layout and format of the page with the aid of CSS.

**Source Explanation**

The assignment contains 8 .php files, 1 CSS file and 1 SQL file. Each .php file represents and page or key functionality of the website, the CSS file is simply for styling the HTML output and the SQL file is to create the entities required for the database. With what we have gathered through the analysis of Nielsen’s Heuristic, we shall now look into the outputted result with that analysis in mind.0

Header.php

The *header.php* file is a default header that is included in every file. It contains a div that acts as a banner for the library. The file also contains PHP code to connect to the relevant data base associated.

Footer.php

*Footer.php* is a script which enacts as the default footer for every page. The file contains code that closes the main <div> of the website along with <body> and <html>.

mainMenu.php

The *mainMenu.php* script contains a *print()* function which will print out html the creates a table of links to the various pages of the site. These pages are the Home page (), the search page and the reservations page. The menu also contains a link to a logout PHP script.

Logout.php

*Logout.php* is a basic script that destroys the user’s current session and redirects them to the login page.

Login.php

The page acts as a basic partition between the user and the core of the website. The page only allows valid users to enter the website, which requires various checks that a user has entered with a valid account. The program begins by checking if a session variables have been set, if so, the user is redirected to the homepage of the site for they have already logged in. The alternative is that a user is presented with a form with that will request a valid username and password. When a user enters their data and submits it, the form posts to the page itself, signalling a check to occur before the form is reprinted as to whether any data was posted to the page. If so, the page will query a SQL statement asking a user exists within the Database exists. If no user exits, a message will appear stating the login failed and the form is printed again. However, if the data is valid the user is sent to the homepage and their username is saved in the SESSION array under ‘user’.

Register.php

*Register.php* is a script that allows a user to create an account to the library to gain access to the site if they do not already have one. When the script begins, it loads in the *header.php* and *mainMenu.php* scripts. Variables that shall be discussed later are initialised to NULL. A check is made whether a post was made to the page. If not, the PHP script shall print out HTML that will present a survey to the user. The registration form contains ten inputs. Once a user fills out there information, a post is made to the PHP page and the aforementioned check is true. All the posted data is stored into the variables that were declared before the check took place. Various other conditions are then checked to see if the user’s information is valid. These checks include whether a user has enter the right number of numbers for a phone number, whether a username is used already, if their entries of the two passwords are incorrect and if there password size is too short. If any of these are met, their data is not inserted into the user’s table within the database and they must re-enter their data accordingly.

Home.php

*Home.php* plays the role of the websites homepage whilst also dealing with any server side activity. The script begins with the inclusion of *Header.php* and *mainMenu.php.* A session is started to access the active session set by *Login.php* or *Register.php*. A check is made to see if a user logged in and declared an element in the *$\_SESSION* array. If not, they are re-directed to the login. A variable entitled *$search* is used to store a string which is a SELECT statement to get the first name of the user. The statement is queried and if successful, the user’s first name is printed.

Once the user name is printed, the script shall then store another SQL statement into *$search* which will return what books the user has reserved from the Reservations table. A variable will then requested the number of rows from the resulting object and shall print the number of books reserved under account which is logged in. After this, the *footer.php* is loaded in and the script is complete. From here, a user will have the option to access the books reserved page, the search books page and the logout page.

searchBook.php

searchBook.php is a .php file that prints a page which will allow a user to search for books on the database. The file begins with the inclusion of *Header.php* and *mainMenu.php*. A check will then be made to ensure that the user logged in with the use of *$\_SESSION[].* If the user did not sign in, they will be redirected to the login page. If not, the user is redirected to the login page. With the use of PHP, a check is made whether a URL parameter known as *offset* is present*.* If so, a variable $offset is initialised to the URL parameter value. If not, the variable is initialised to zero.

A SQL statement is stored into a string variable *$catSearch* to search for all category names. The names are fetch by the while loop from the returned object from the SQL query. Following this, an array of conditions are made. First off, a condition is made to check whether a user clicked reserve on a book. If so, various statements will be queried to update and insert relevant data into the Book and Reservations tale. Each statement is met with a check via the else’s of the nested if statements. Once all the relevant information is stored, the page will inform whether the book was reserved successfully or not.

Whether a book was clicked reserve or not, a form is outputted via PHP and is formatted with CSS. The form is a search form for the books in the library. The form contains two text input boxes for the book title and author whilst a select menu is created for the categories. As mentioned previously, all category names are saved in an array. While the PHP prints the html, a loop occurs within the <select> element and creates options for the categories. Each options value is determined by a counter that is incremented after each iteration of the loop.

If a user submits a search query, the form posts to the page itself and all processes mentioned previously occur. However, after the printing of the search table via PHP, a check that determines a post is made by the values passed via URL parameters is considered to be true. A sql statement is then created that selects all data from the books table where the book title, author and category that is similar to the data passed via the search. The first statement queried will find how many rows there are in the table that match this request, the second query shall carry out the same action, although now with the use of *LIMIT OFFSET*. *LIMIT* will only allow the database to return a certain number of rows whilst *OFFSET* acts as an index for the statement. For instance, if the LIMIT is 5 and the *OFFSET* is 1, the second row to the sixth row will be returned if that many exist. If the $offset variable is less than the number of total rows than calculated previously, than a next button shall appear, allowing a user to see the next 5 rows of data. This is achieved by passing the page link with a URL parameter that is the *$offset + 5*. When the link is clicked, the GET offset check is utilised to save the users position. If a user’s offset is 5 or greater, a *previous* link will display allowing a user to move back to the previously display rows of data.

reserveBook.php

The *reserveBook.php* page begins with the loading of the menu and header with the use of the include() function within PHP while a check is also made to see if a valid session was created by the user to allow them access to the contents of said page. The same process for the loading and storing of category names that took place in the *searchBook.php* script occurs again. After this, a check is made to see whether the user submitted a form to the page. If false, a table is printed that displays a user’s reserved books. This is achieved by creating a SQL statement that selects required data from the books table and the Reservations table. This connection between two tables utilises their foreign key to each other to create an inner join. A table is then outputted by printing html within a PHP print statement.

Within each row, there is a column that, when clicked, shall unreserved the book associated in that row of data. As mentioned previously, when the script begins, there is a check that checks whether a book is being unreserved or not. What occurs is that the link, when clicked, reloads the page but adds a URL parameter entitled *$unReserved* which will contain the books unique ISBN code. Since the statement is now true SQL statements shall be queried through nested if conditions to delete the relevant data in terms of removing the reservation attached to the book. Therefore, DELETE statements are queried to the Reservations table and UPDATE statements are made to the book table. Once all is updated according, a message will appear on the page informing the user the update was a success. The table printed will either display a message that there are no books reserved by the user or it shall print the updated table. The number of rows displayed is similar to the *OFFSET LIMIT* used in *searchBook.php*.

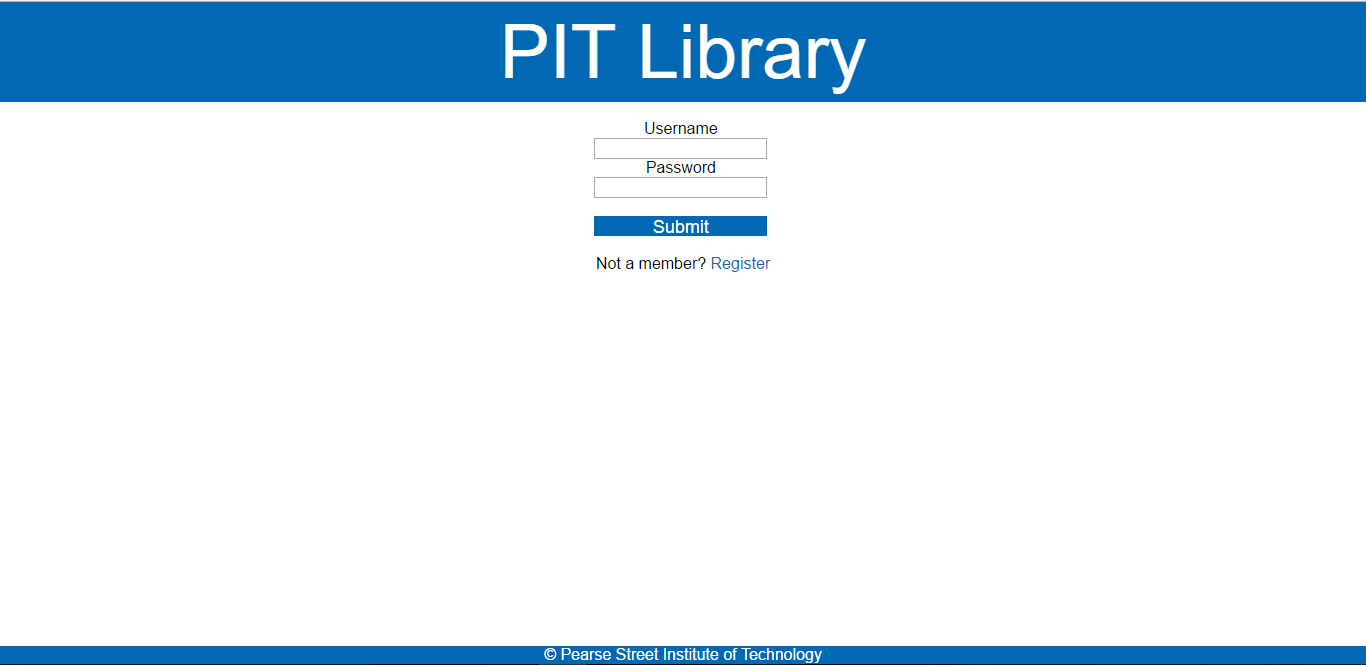
format.css

The format.css file is utilised to style the website. It contains classes and id’s which are loaded by outputted html.

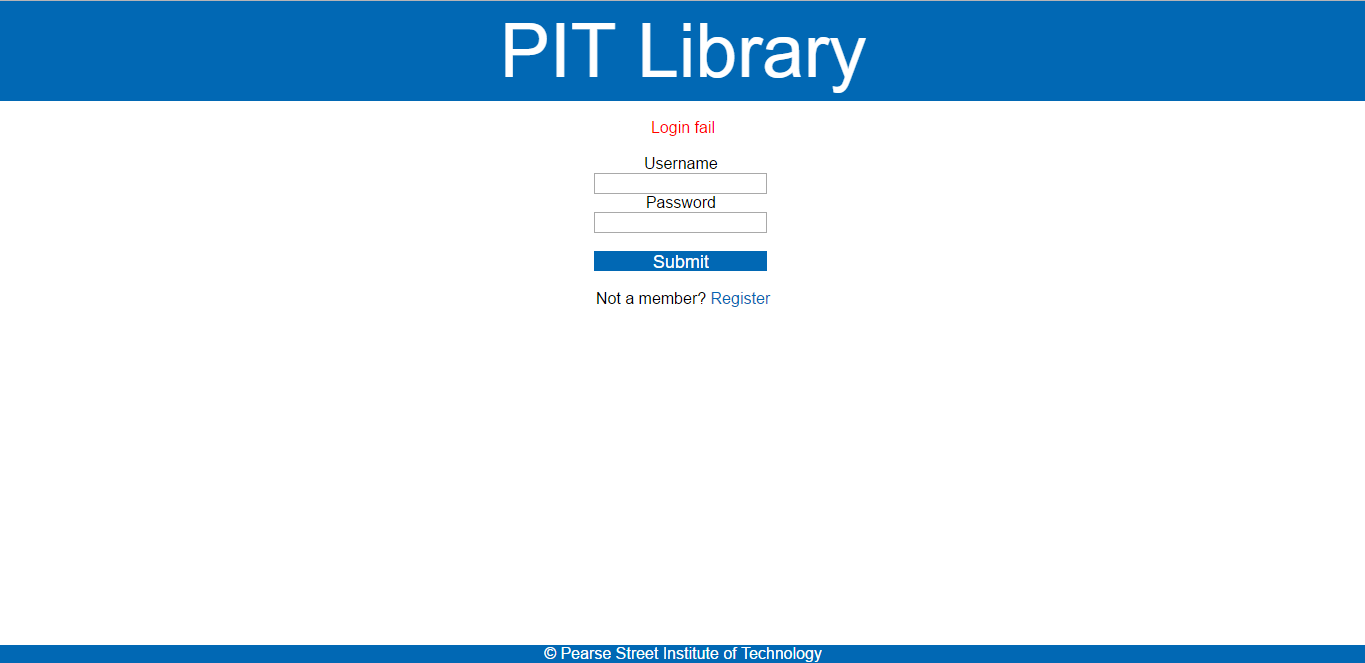
assignment.sql

Creates tables with attributes that are used to store data about the library. Four tables are utilised and INSERT statements are present to load default data into the entities.

**User-End Experience**

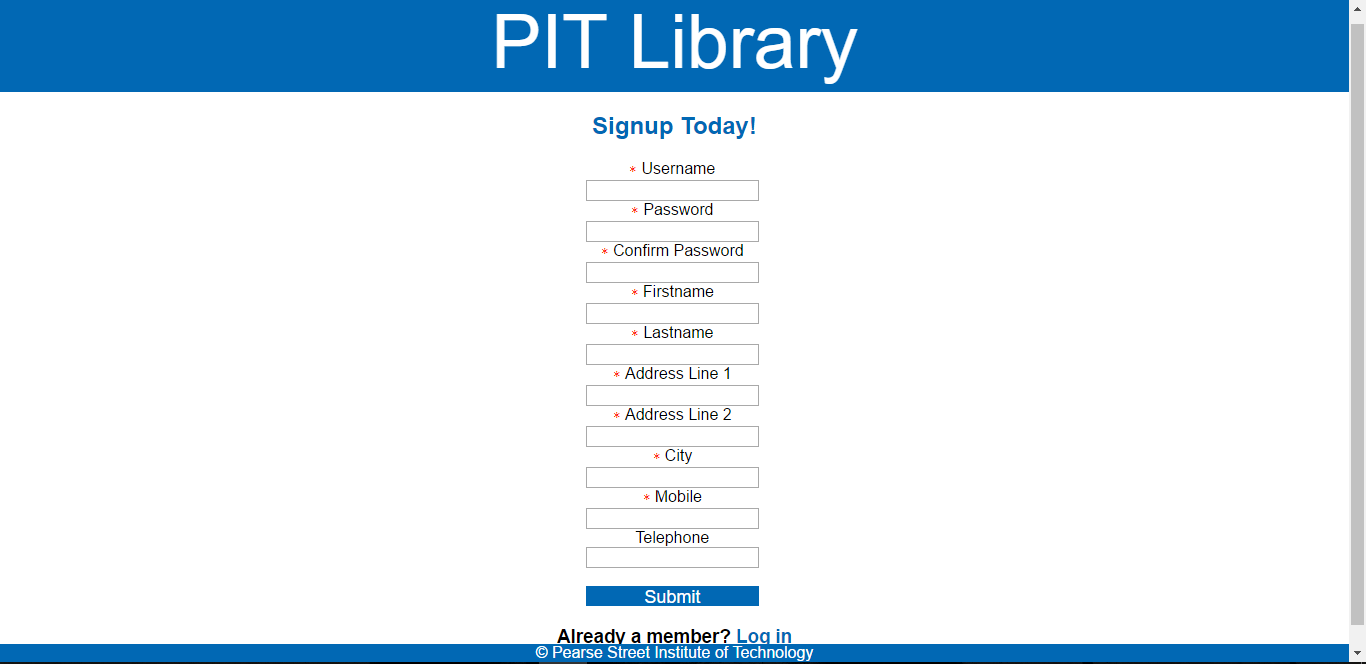
Login.php 

When a user visits the log in page, they will meet the default header and footer along with a form requesting their user credentials.

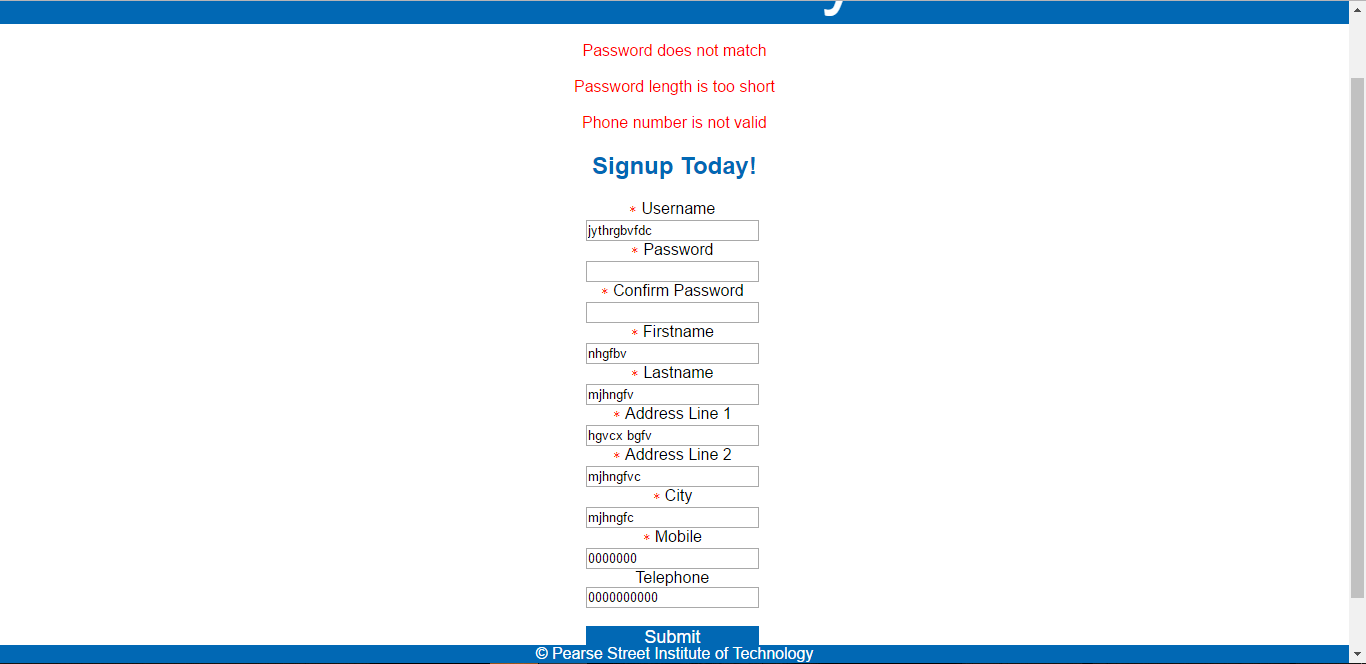


As seen with the above screen shot, if a users credentials are not valid, a warning message shall appear informing them of a *Login Fail* in a red font. If a user has no account, they are given the option to register an account.

Register.php

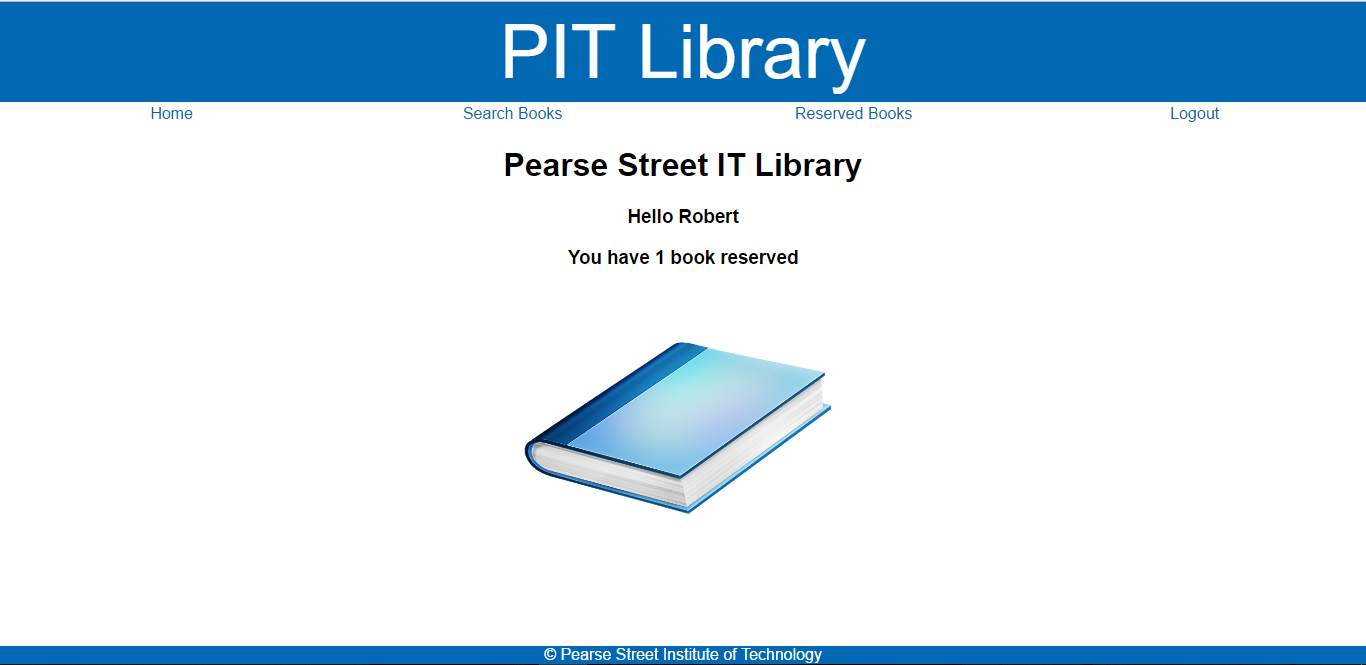


The Register.php page once again presents a form to the user requesting them to enter information in order to setup a library account. As may discover in both the Login and Register page is that the submit button features a design that is similar to the overall page. This is achieved with the use of CSS. Any information that is required is marked with a red star.



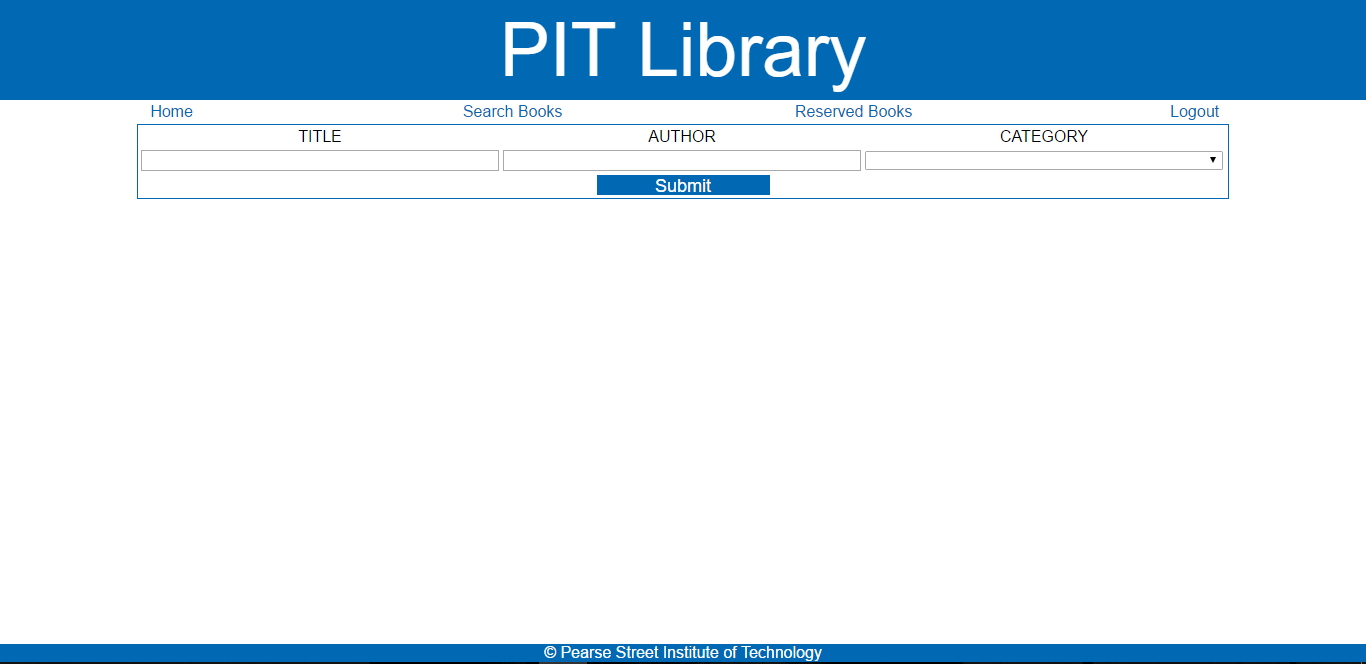
As seen in the above screenshot, appropriate error handling is encountered if the user has submitted inaccurate data. Each error the user made is displayed the form informing them as to why their account request was denied. If a user’s request is denied, the data they previously entered is saved in the input fields to allow them to edit there previously entered data with the requested corrections in mind.

Home.php

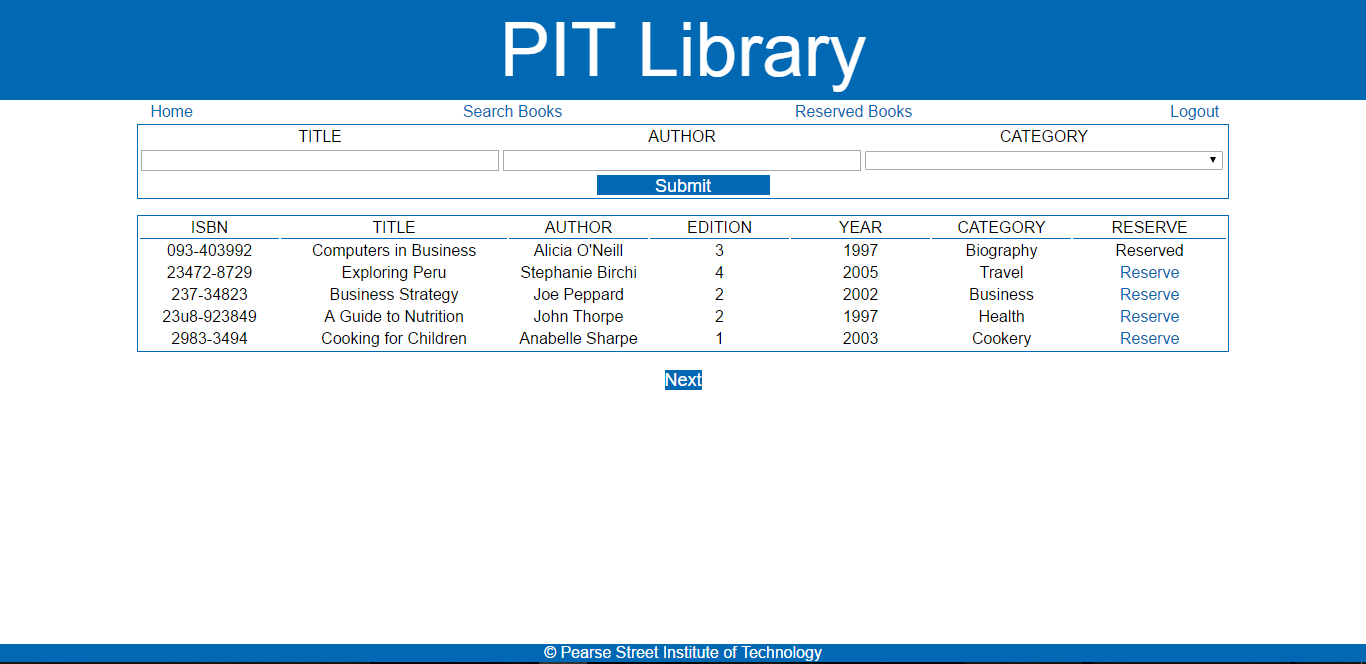


Once the user is logged in, they will be directed to the homepage. The homepage contains greets the user by their first name and informs said user as to how many books they have reserved under their account. A navigation menu is also present under the pages header to allow a user to navigate through the different pages in the site.

searchBooks.php

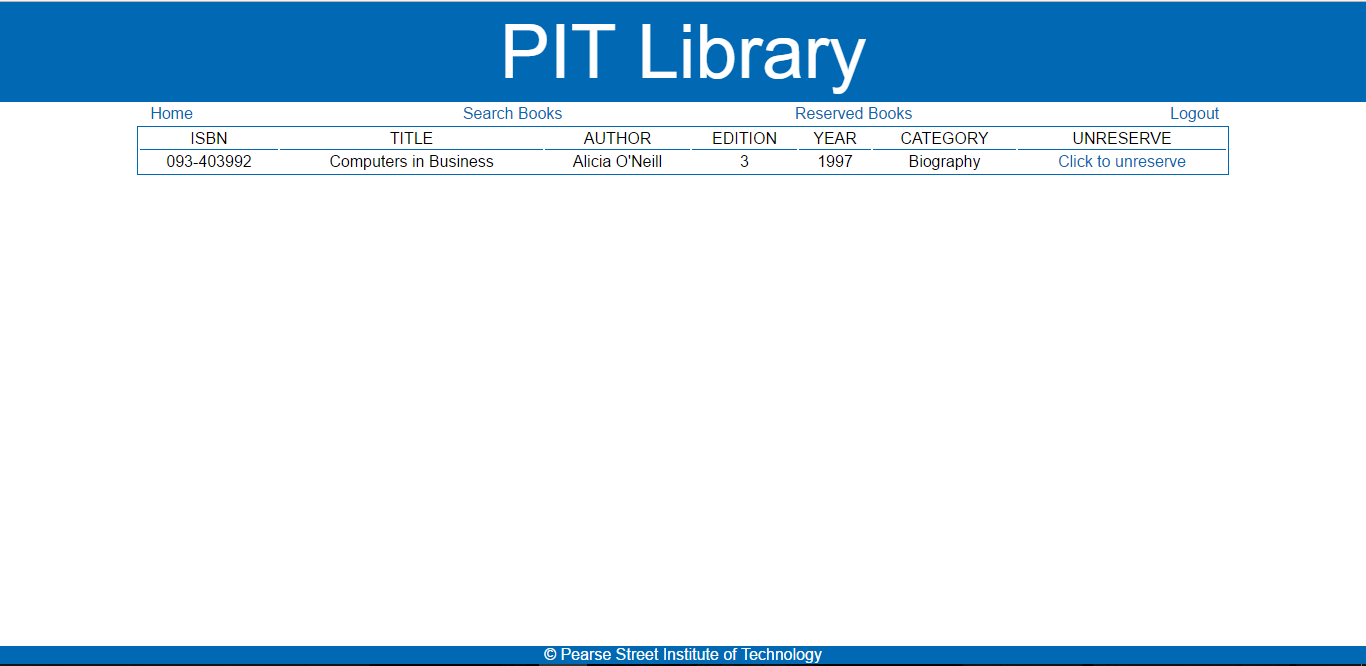


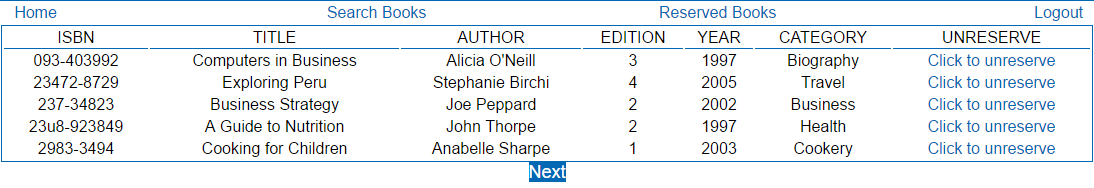
The search page within the site presents the user with a basic form which allows them to search for a book by title, author and/or category.



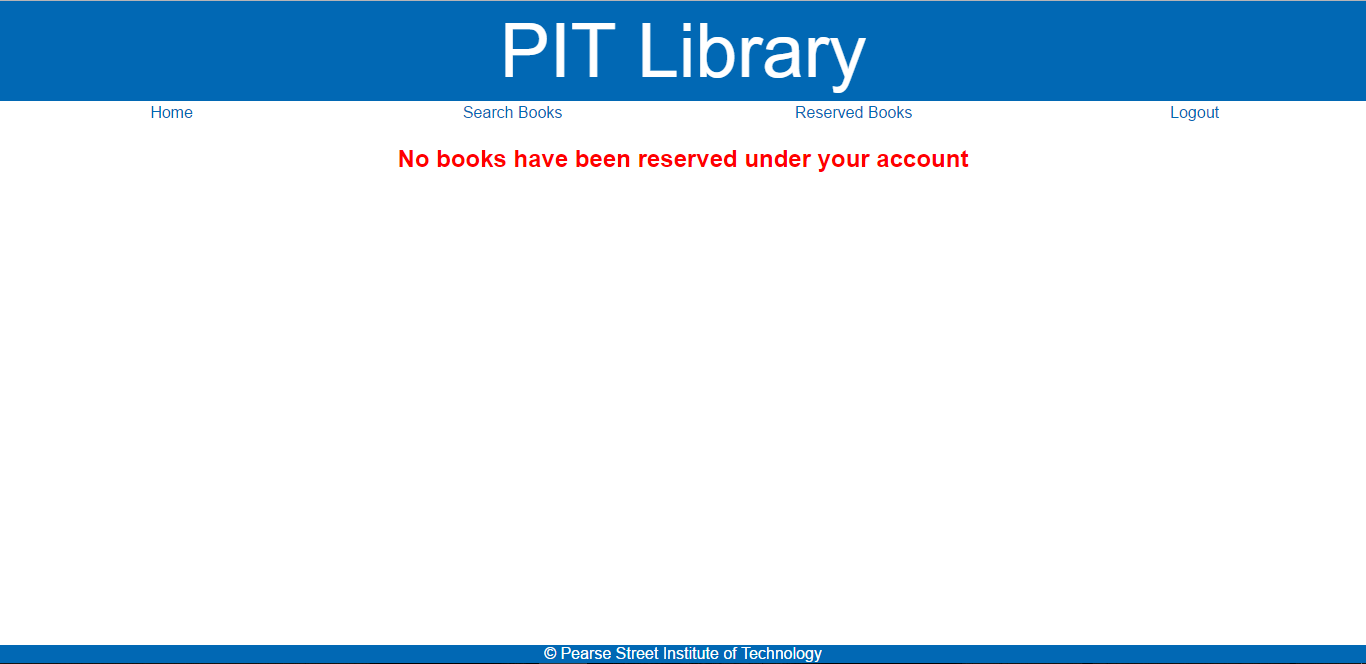
As seen above, if a user’s search is valid, a list of books is displayed with some information on said books. Only 5 books can be displayed on a page as one time. The user has an option to view the next 5 books (if there are anymore) by clicking the *Next* button below the results. When they view the next list, an option known as *Prev* shall appear, allowing one to go back to their previous results.

reserveBooks.php





Above is a screenshot of the “reserved list” page. This page will present to the user what books they have reserved with the use of a table. If a user no longer wants to reserve said book, they have the option to click a link next to the books information. This link, once pressed shall remove the user’s association to that book. Similar to the search page, only 5 rows of data can be displayed at the one time. Therefore, if a user has over five books and wishes to view the next set of books, they will be presented with a link to view the next set of data. If no books are present, a message will appear to the user stating they have no reservations.



If a user clicks the logout link on the navigation bar, they will simply be signed out of their account and redirected to the login page.

**List of Source Code**

* Header.php
* Footer.php
* mainMenu.php
* Login.php
* Logout.php
* Register.php
* Home.php
* searchBooks.php
* reserveBooks.php
* format.css
* assignment.sql