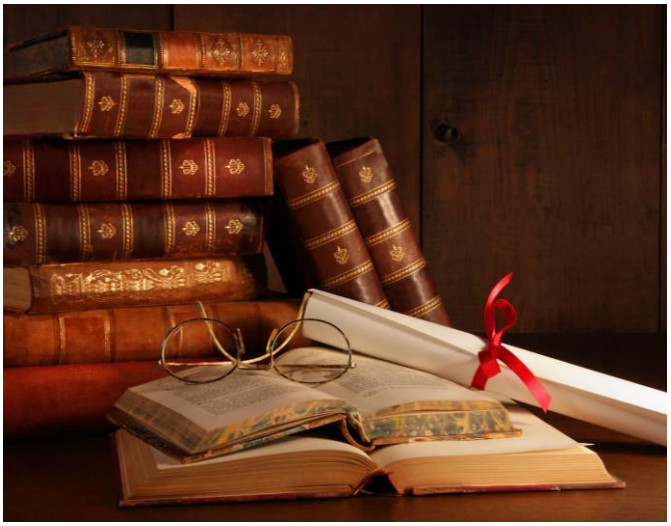


BOOKSTORE

Web Services

NAME

Beeharry Mohammad Taajuddin (*2003_19375*)

COURSE

BSc (Hons) Software Engineering

COHORT

Year 2 Semester 2

Lecturer

Dr. G. Suddul

Table of Contents

List of figures 2

Bookstore Design Client Side 4

Homepage (index.html) 4

Add Book (addbook.html) 5

Search Book 6

By title (searchbooktitle.html) 6

By author (searchbookauthor.html) 7

By publication year (searchbookyear.html) 9

By category (searchbookcat.html) 10

Advanced Book 11

By author & category (advancedsearchAC.html) 11

By author & publication year (advancedsearchAY.html) 13

View All Books (viewallbooks.html) 15

Book Format (bookformat.html) 16

Implementation 17

bookFunctions.php (Server Side) 17

index.php (Server Side) 23

Add Book 25

Search Book 26

By title 26

By author 27

By publication year 28

By category 29

Advanced Book 30

By author & category 30

By author & year 31

View All Book 32

Book Format 34

Search by author & publication year 36

Search by author & title 37

Search by title & publication year 38

Testing 39

Add Book 39

Search Book 40

By title 40

By author 41
By publication year 42
By category 43
Advanced Search Book 44
By author & category 44
By author & publication year 45
View All Book 46
Book Format 48

List of figures

Figure 1- Homepage (index.html) 4
Figure 2 - Add Book (addbook.html) 5
Figure 3 - Search by title (searchbooktitle.html(1)) 6
Figure 4 - Search by title (searchbooktitle.html(2)) 7
Figure 5 - Search by author (searchbookauthor.html(1)) 7
Figure 6 - Search by author (searchbookauthor.html(2)) 8
Figure 7 - Search by pub year (searchbookyear.html(1)) 9
Figure 8 - Search by pub year (searchbookyear.html(2)) 9
Figure 9 - Search by category (searchbookcat.html(1)) 10
Figure 10 - Search by category (searchbookcat.html(2)) 10
Figure 11 - Advanced search by author & category (advancedsearchAC.html(1)) 11
Figure 12 - Advanced search by author & category (advancedsearchAC.html(2)) 12
Figure 13 - Advanced search by author & pub year (advancedsearchAY.html(1)) 13
Figure 14 - Advanced search by author & pub year (advancedsearchAY.html(2)) 14
Figure 15 - View all books (viewallbooks.html) 15
Figure 16 - Book format (bookformat.html) 16
Figure 17 - Json book format 16
Figure 18 - Xml book format 17
Figure 19 - Get book by title, author (bookFunctions.php) 17
Figure 20 - Get book by year, category (bookFunctions.php) 18
Figure 21 - Get book by author & category, author & year (bookFunctions.php) 19
Figure 22 - Get all books (bookFunctions.php) 19
Figure 23 - Add book (bookFunctions.php) 20
Figure 24 - Get book format (bookFunctions.php(1)) 21
Figure 25 - Get book format (bookFunctions.php(2)) 22
Figure 26 - index.php (1) 23
Figure 27 - index.php (2) 24
Figure 28 - Add book Server Side (addbook.php) 25
Figure 29 - Add book Client Side (addbook.html) 25
Figure 30 - Add book script Client Side (addbook.html) 26
Figure 31 - Search book by title (searchbooktitle.html) 26
Figure 32 - Search book by title script (searchbooktitle.html) 27
Figure 33 - Search book by author (searchbookauthor.html) 27
Figure 34 - Search book by author script (searchbookauthor.html) 28

Figure 35 - Search book by pub year (searchbookyear.html) 28
Figure 36 - Search book by pub year script (searchbookyear.html) 29
Figure 37 - Search book by category (searchbookcat.html) 29
Figure 38 - Search book by category script (searchbookcat.html) 30
Figure 39 - Search book by author & category (advancedsearchAC.html) 30
Figure 40 - Search book by author & category script (advancedsearchAC.html) 31
Figure 41 - Search book by author & pub year (advancedsearchAY.html) 31
Figure 42 - Search book by author & pub year script (advancedsearchAY.html) 32
Figure 43 - View all books (viewallbooks.html) 32
Figure 44 - View all books script (viewallbooks.html) 33
Figure 45 - bookformat.php (1) 34
Figure 46 - bookformat.php (2) 35
Figure 47 - bookformat.php (3) 36
Figure 48 - Book format author & pub year (bookformat.html) 36
Figure 49 - Book format author & pub year script (bookformat.html) 37
Figure 50 - Book format author & title (bookformat.html) 37
Figure 51 - Book format author & title script (bookformat.html) 37
Figure 52 - Book format title & pub year (bookformat.html) 38
Figure 53 - Book format author & title script (bookformat.html) 38
Figure 54 - Testing book format author & pub year in json 48
Figure 55 - Testing book format author & title in xml 48
Figure 56 - Testing book format title & pub year in json 48

Bookstore Design Client Side

Homepage (index.html)

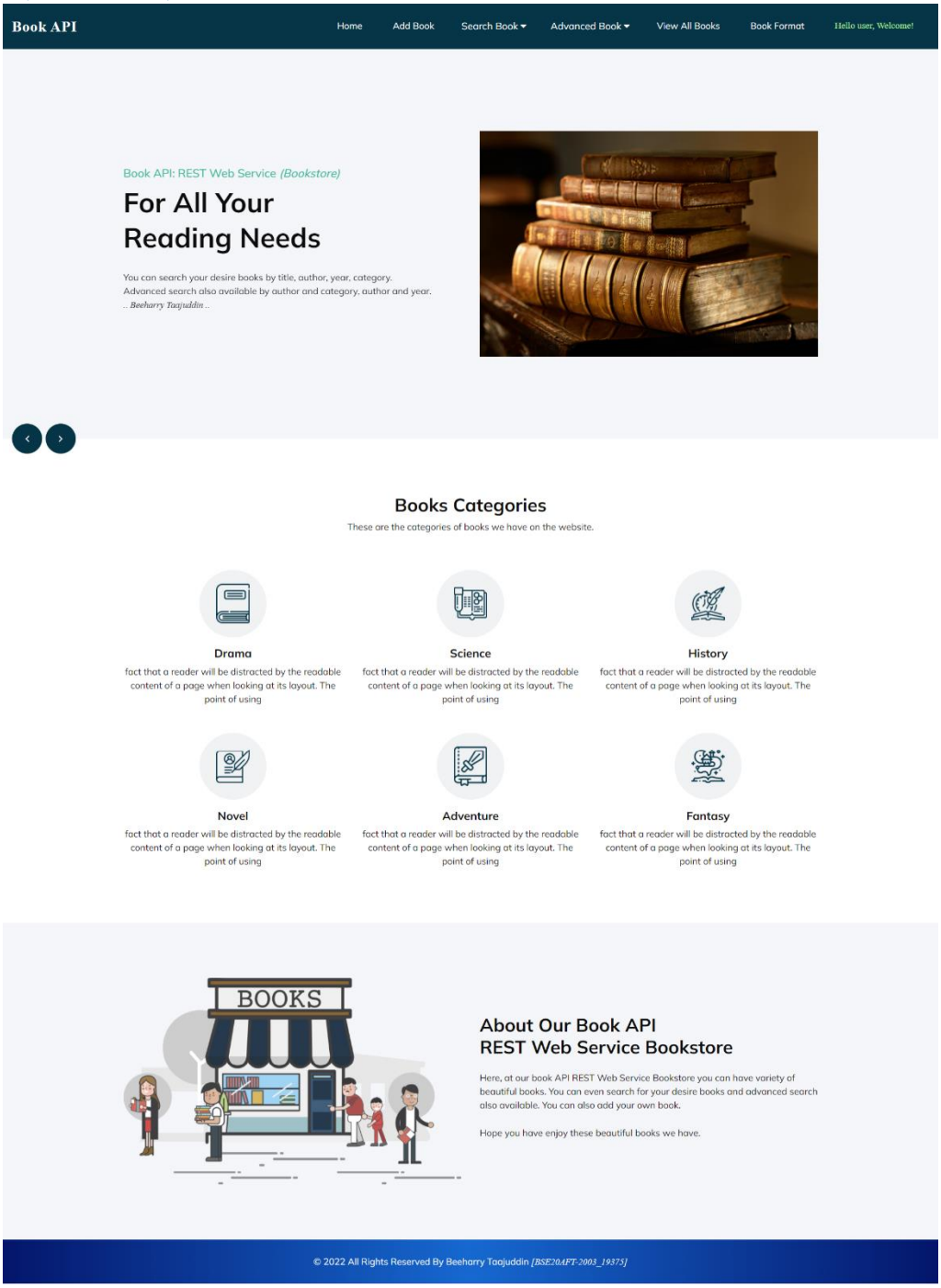


Figure 1- Homepage (index.html)

The home page is depicted in Figure 1. It is user friendly and user can go through other page to search for books under the dropdown **Search Book** and **Advanced Book**. Furthermore, users can view all available books in the bookstore and also add a book. A web service has been implemented for programmers use.

Add Book (addbook.html)

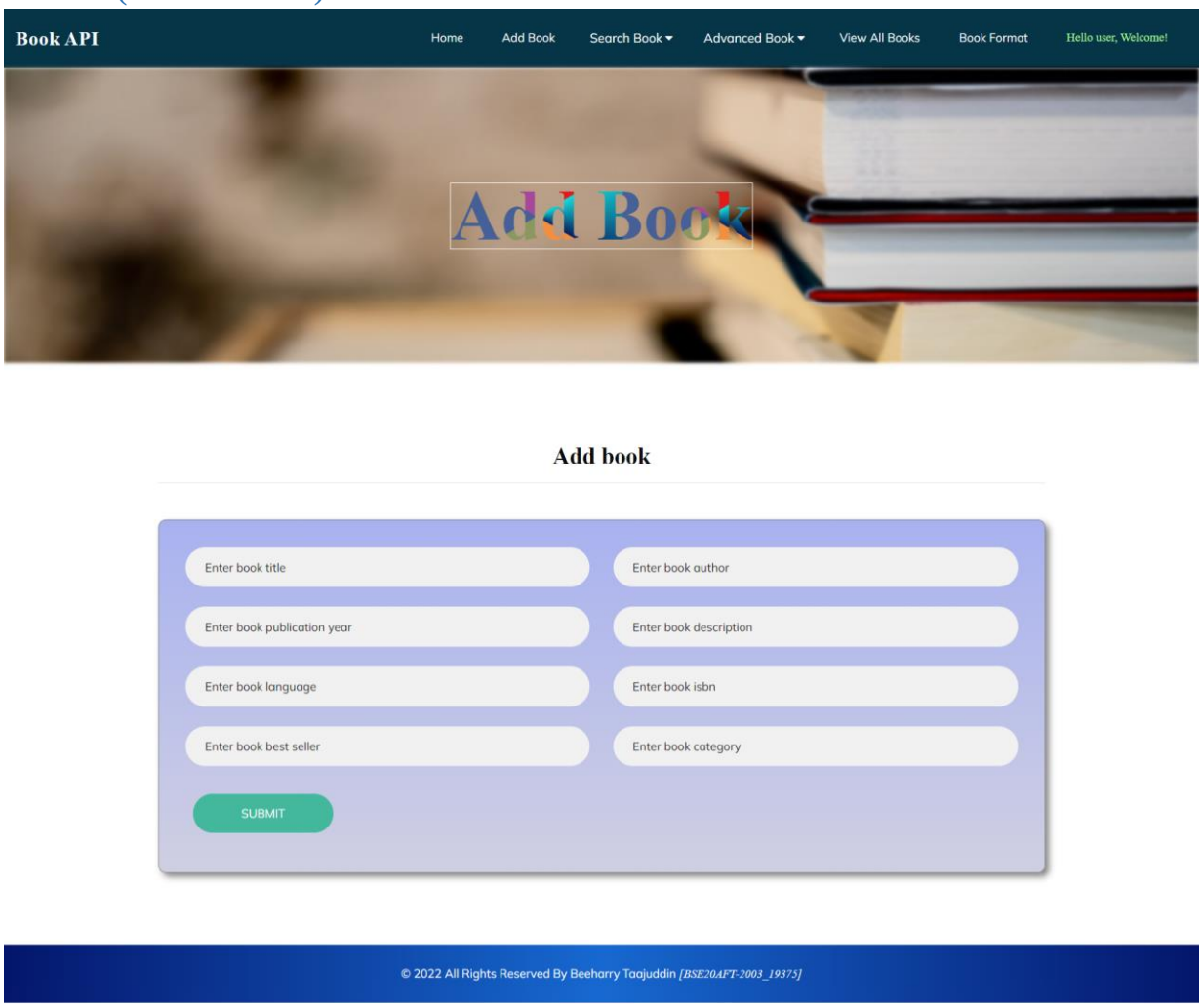


Figure 2 - Add Book (addbook.html)

Figure 2 depicts the add book page. If ever an author wants to publish his/her book, he/she simple navigate to the add book page.

Search Book

If a user wants to search for books by the book's title, author, publication year and category, a dropdown **Search Book** has been implemented.

By title (searchbooktitle.html)

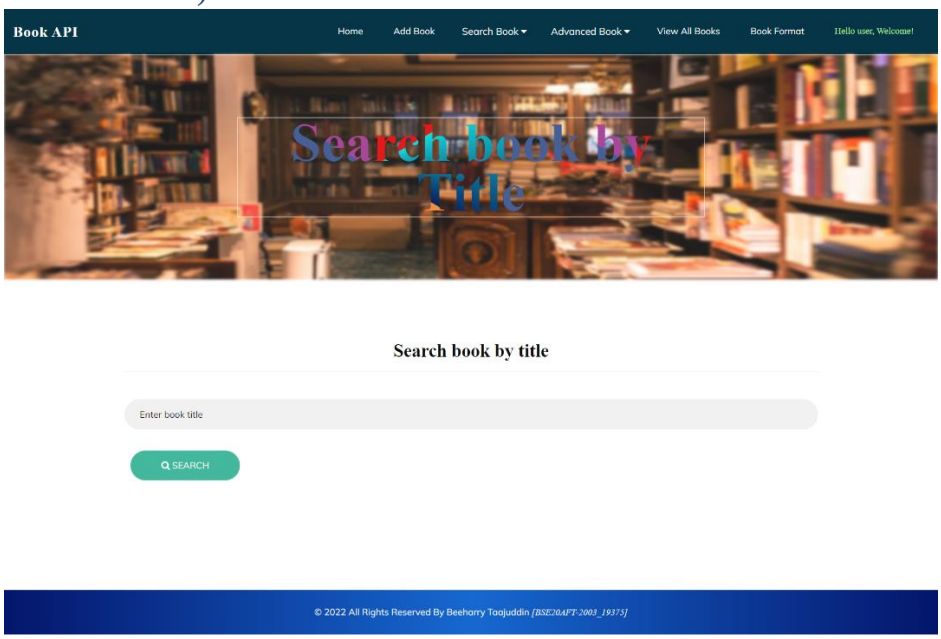


Figure 3 - Search by title (searchbooktitle.html(1))

Figure 3 shows the user interface of the search book by title page with a textbox and a search button. Users can type any starting letters to perform the search.

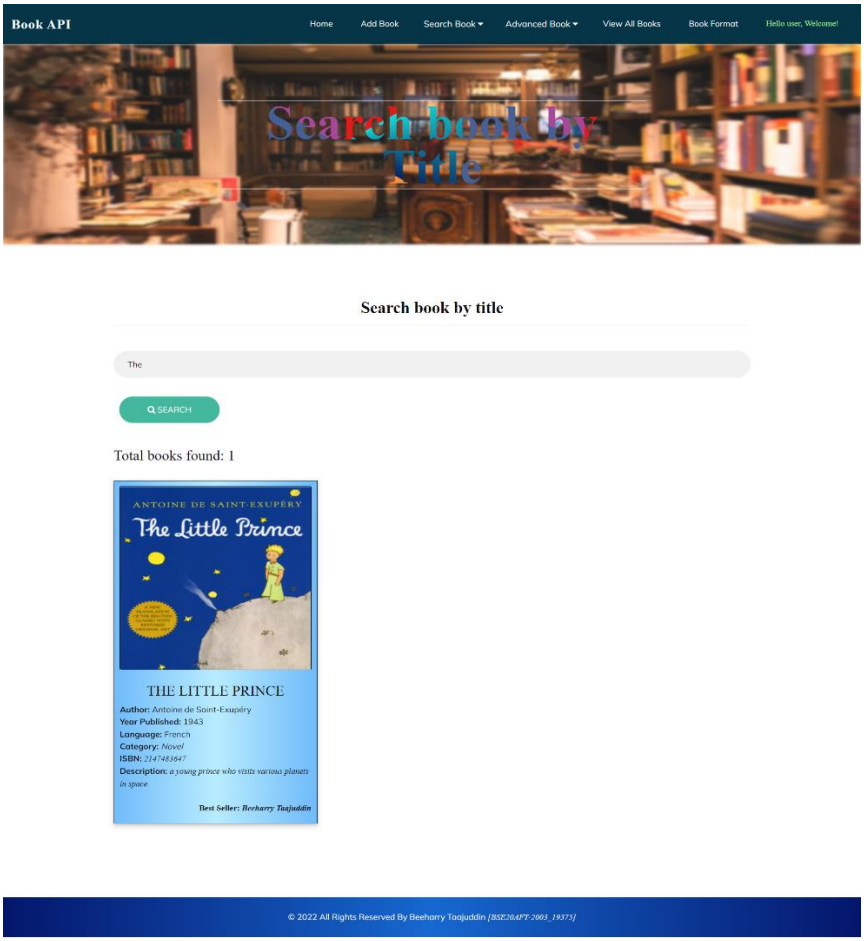


Figure 4 - Search by title (searchbooktitle.html(2))

Figure 4 shows the result for the starting letters and it displays the total number of book found in the bookstore along with the book and description.

By author (searchbookauthor.html)

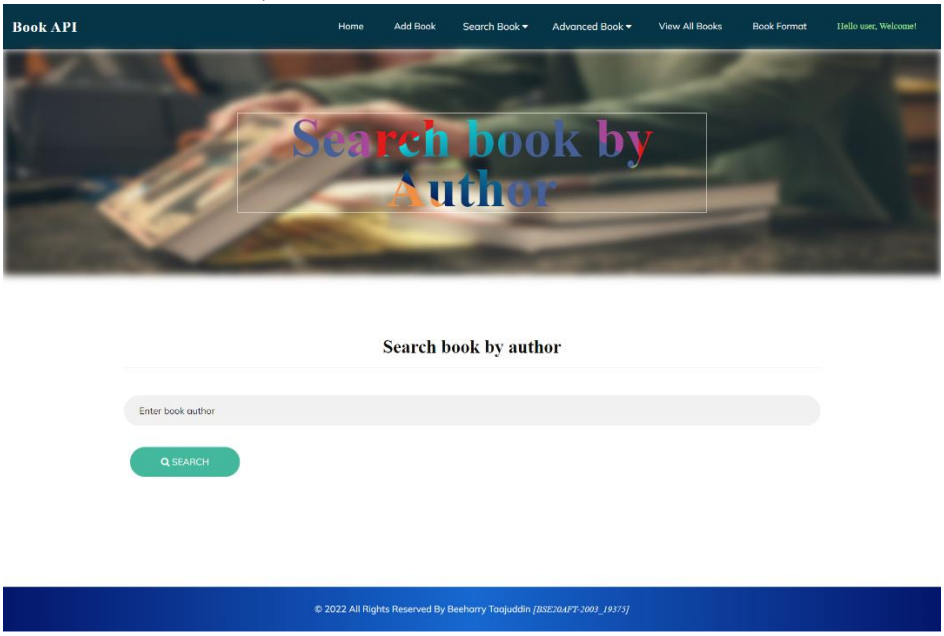


Figure 5 - Search by author (searchbookauthor.html(1))

Figure 5 shows the UI of the search author page. Users can type any starting letters to perform the search.

Figure 6 - Search by author (searchbookauthor.html(2))

In figure 6, the search by the author name displays the book and also the total number of book found in the bookstore.

By publication year (searchbookyear.html)

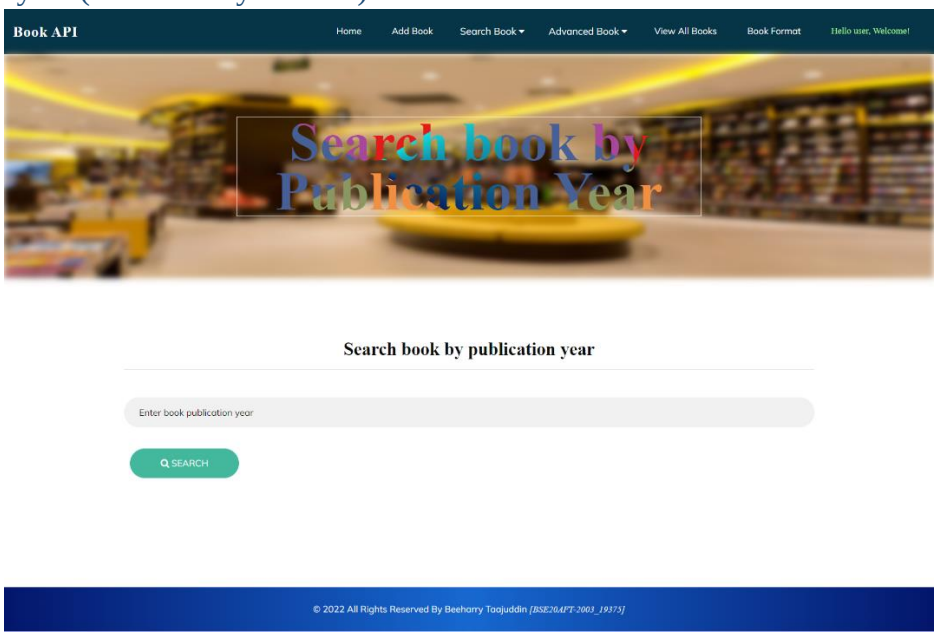


Figure 7 - Search by pub year (searchbookyear.html(1))

Figure 7 shows the search book by publication year page. Users can type any starting numbers to perform the search.

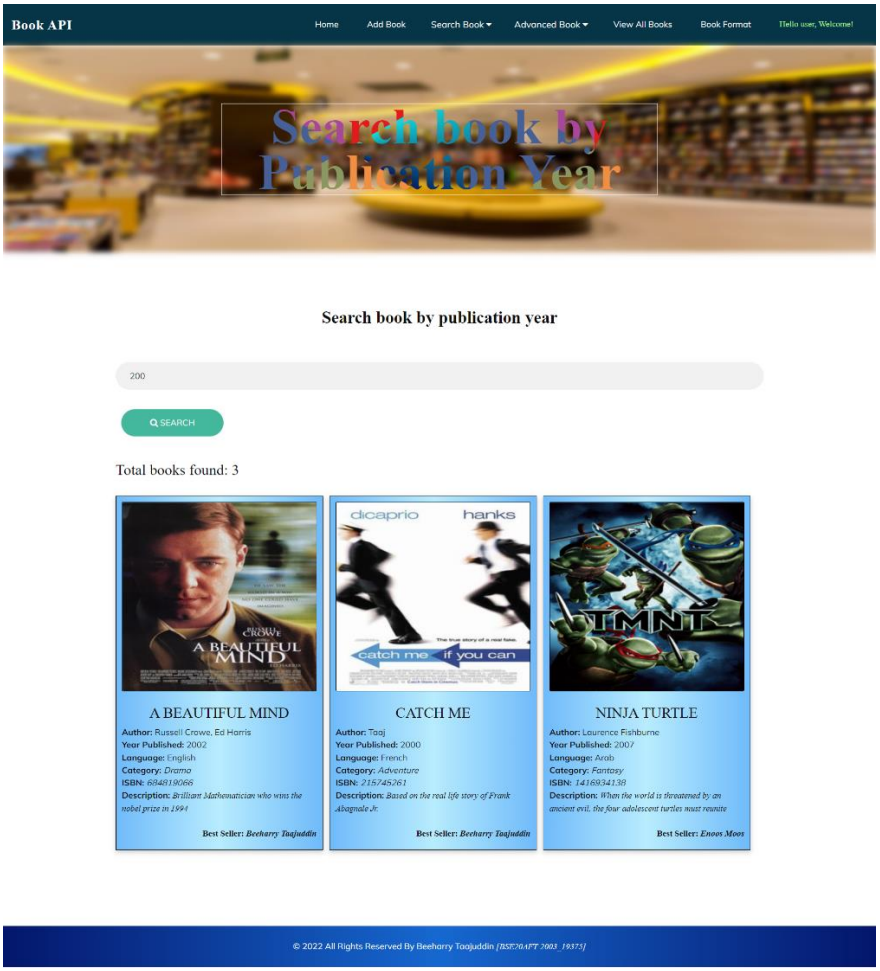


Figure 8 - Search by pub year (searchbookyear.html(2))

In figure 8, the user has performed a search by 3 digits and it displays the total number of book along with the book which have the 3 digits in their publication year.

By category (searchbookcat.html)

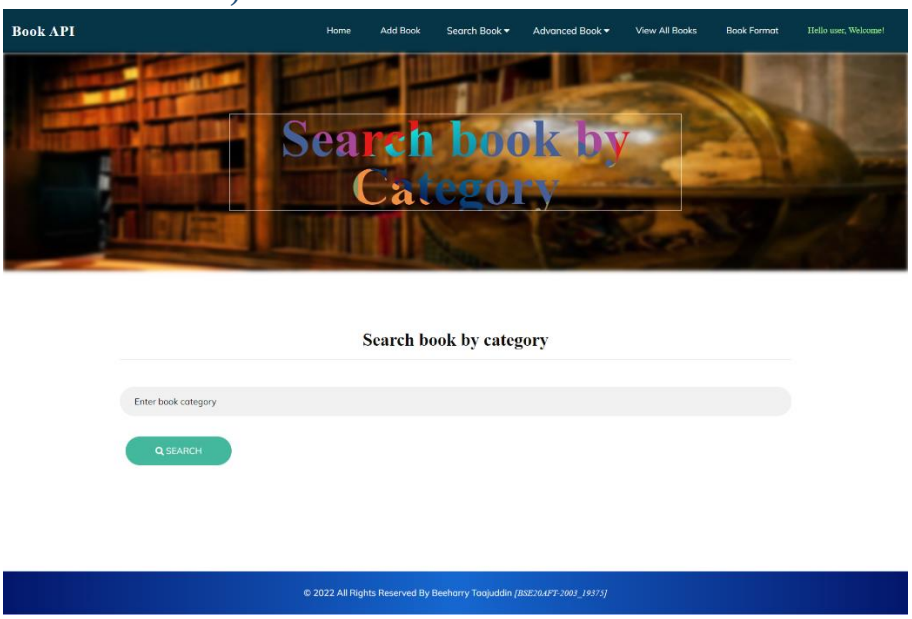


Figure 9 - Search by category (searchbookcat.html(1))

Figure 9 shows the search book by the category page where user can search for book starting letters.

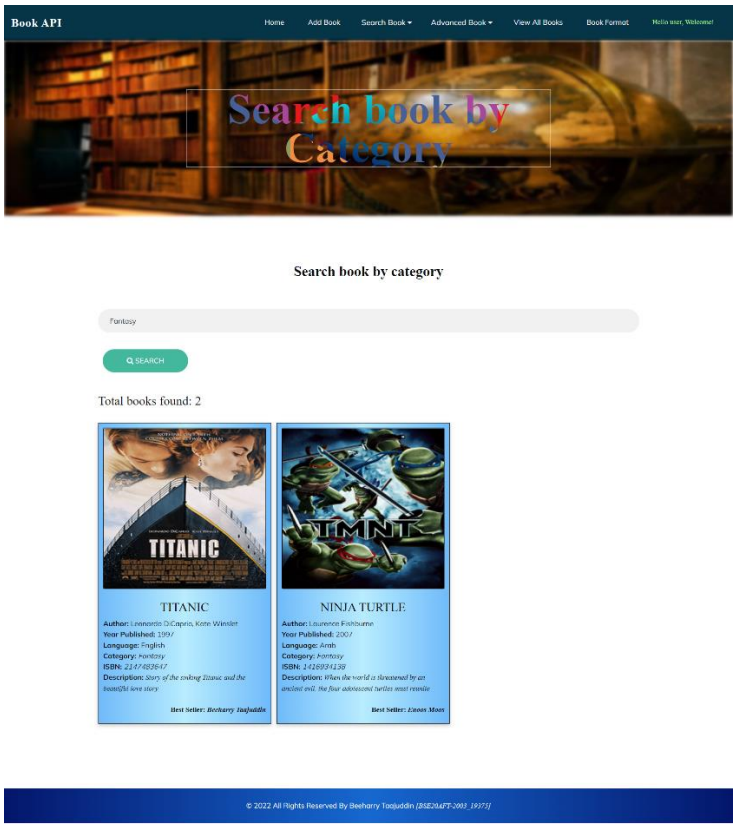


Figure 10 - Search by category (searchbookcat.html(2))

Figure 10 shows that a user has search a book category by *Fantasy* and as result, it displays the total number of book found in the bookstore along with the book and description.

Advanced Book

Whenever a user wants to search a book, which means *advanced search*, by author and category or author and year, he/she simply need to hover on the **Advanced Book**.

By author & category (advancedsearchAC.html)

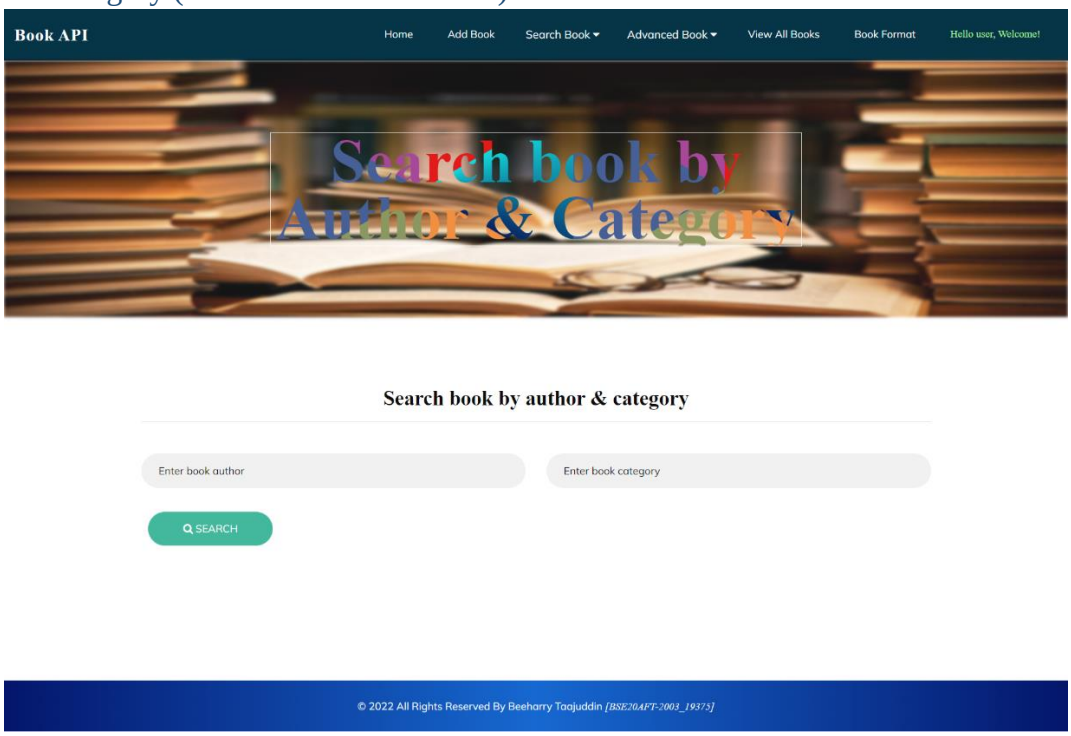


Figure 11 - Advanced search by author & category (advancedsearchAC.html(1))

Figure 11 shows the advanced search page for author and category. User can perform this activity by the starting letters for both search.

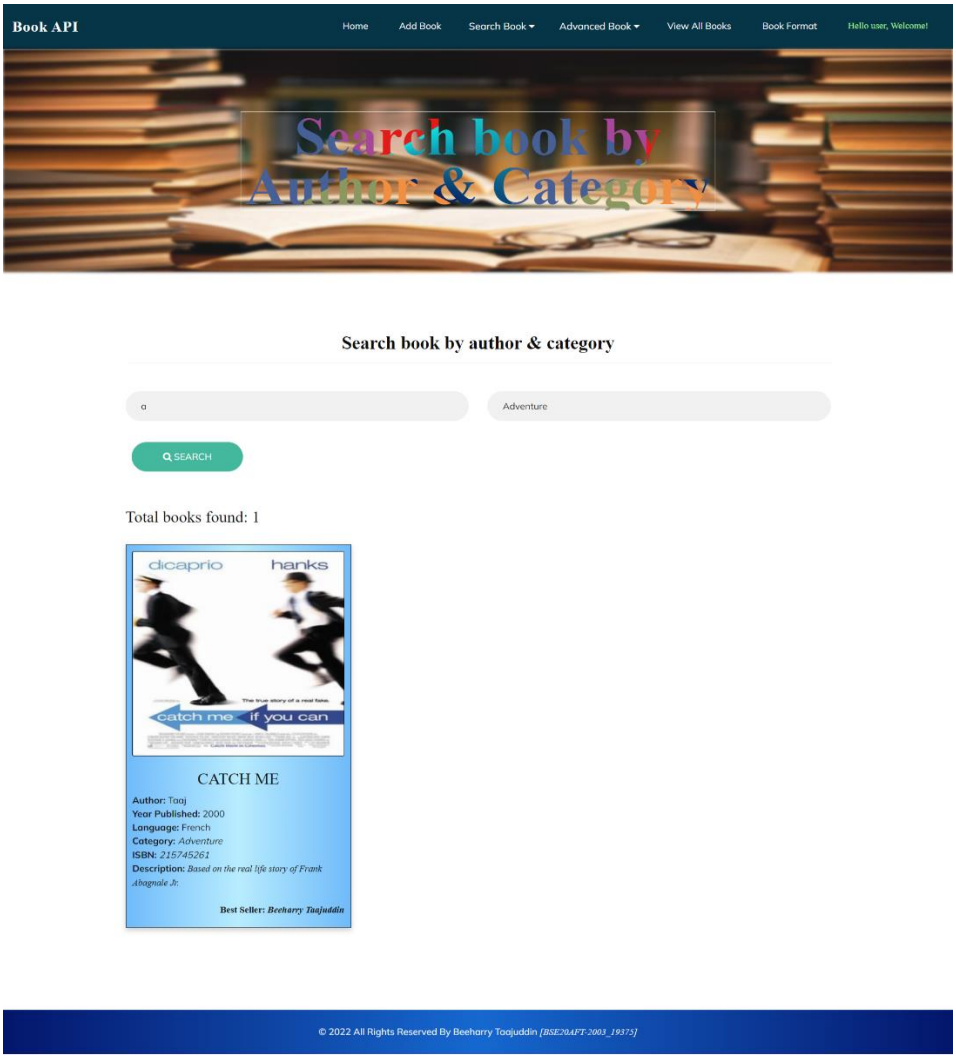


Figure 12 - Advanced search by author & category (advancedsearchAC.html(2))

In figure 12, as results, the available books in the bookstore are displayed and also the number of book found.

By author & publication year (advancedsearchAY.html)

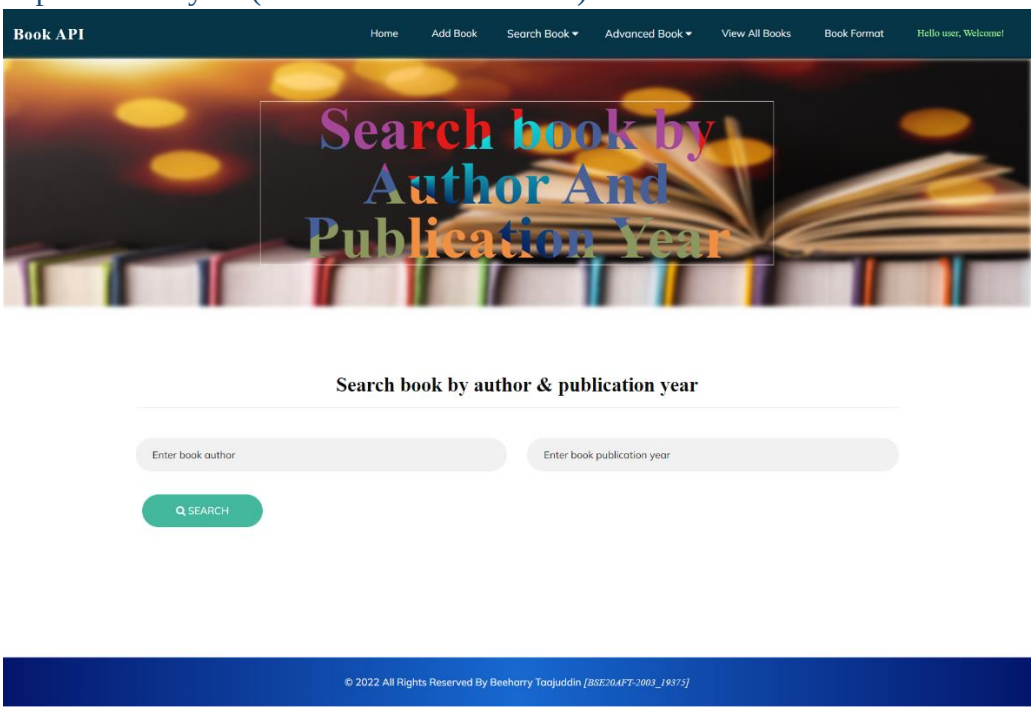


Figure 13 - Advanced search by author & pub year (advancedsearchAY.html(1))

Figure 13 shows the advanced search page for author and publication year. User can perform this activity by the starting letters and digits.

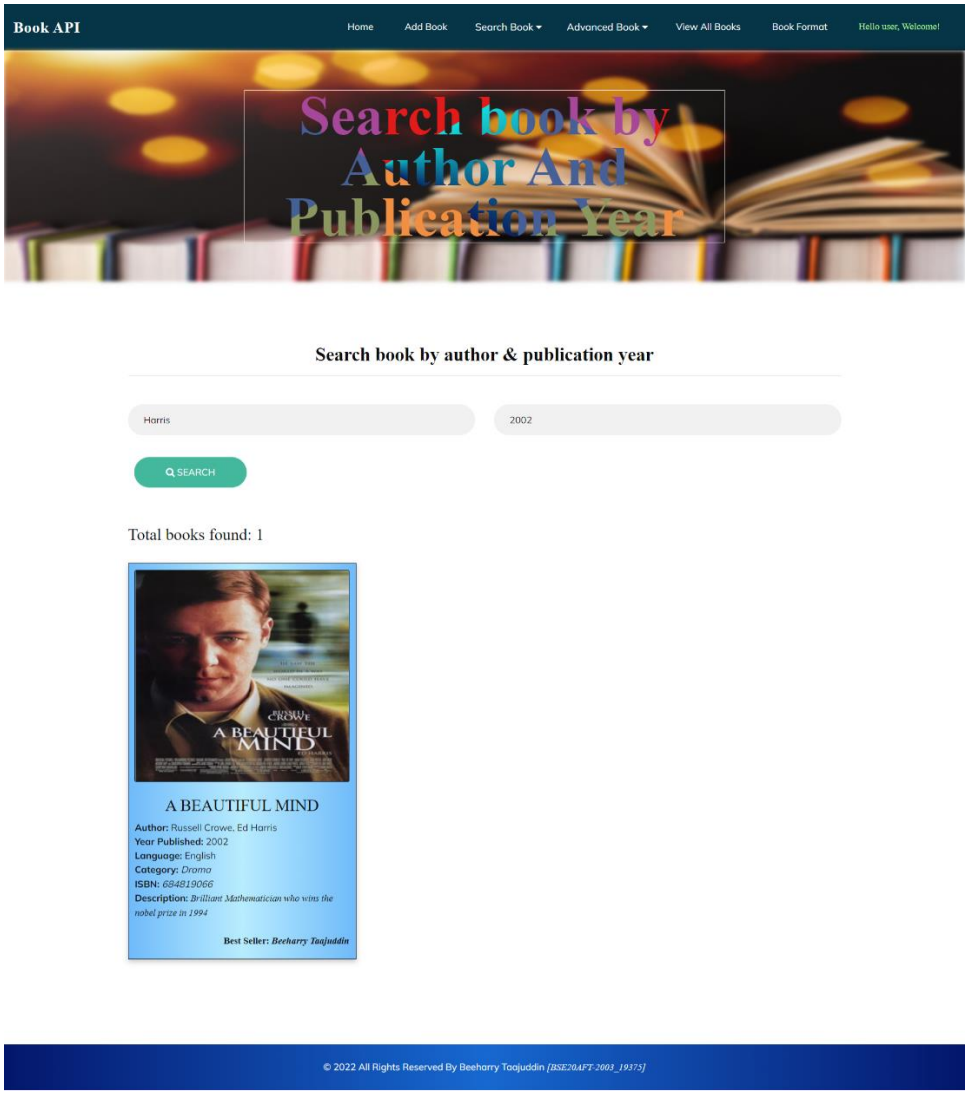


Figure 14 - Advanced search by author & pub year (advancedsearchAY.html(2))

In figure 14, as results, the available books in the bookstore are displayed and also the number of book found.

View All Books (viewallbooks.html)

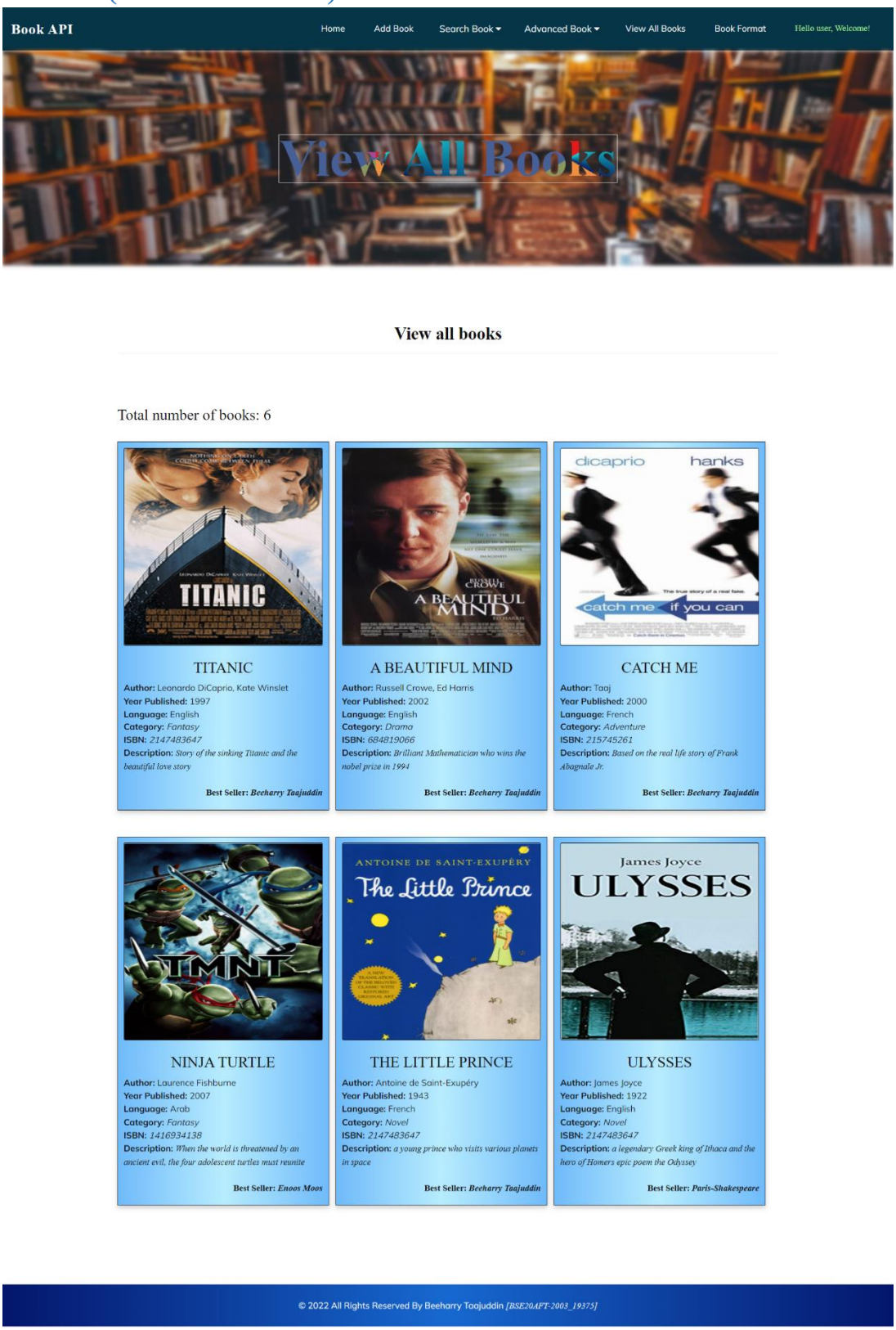


Figure 15 - View all books (viewallbooks.html)

In figure 15, users can view the amount of books available and also list of books with their description in the bookstore.

Book Format (bookformat.html)

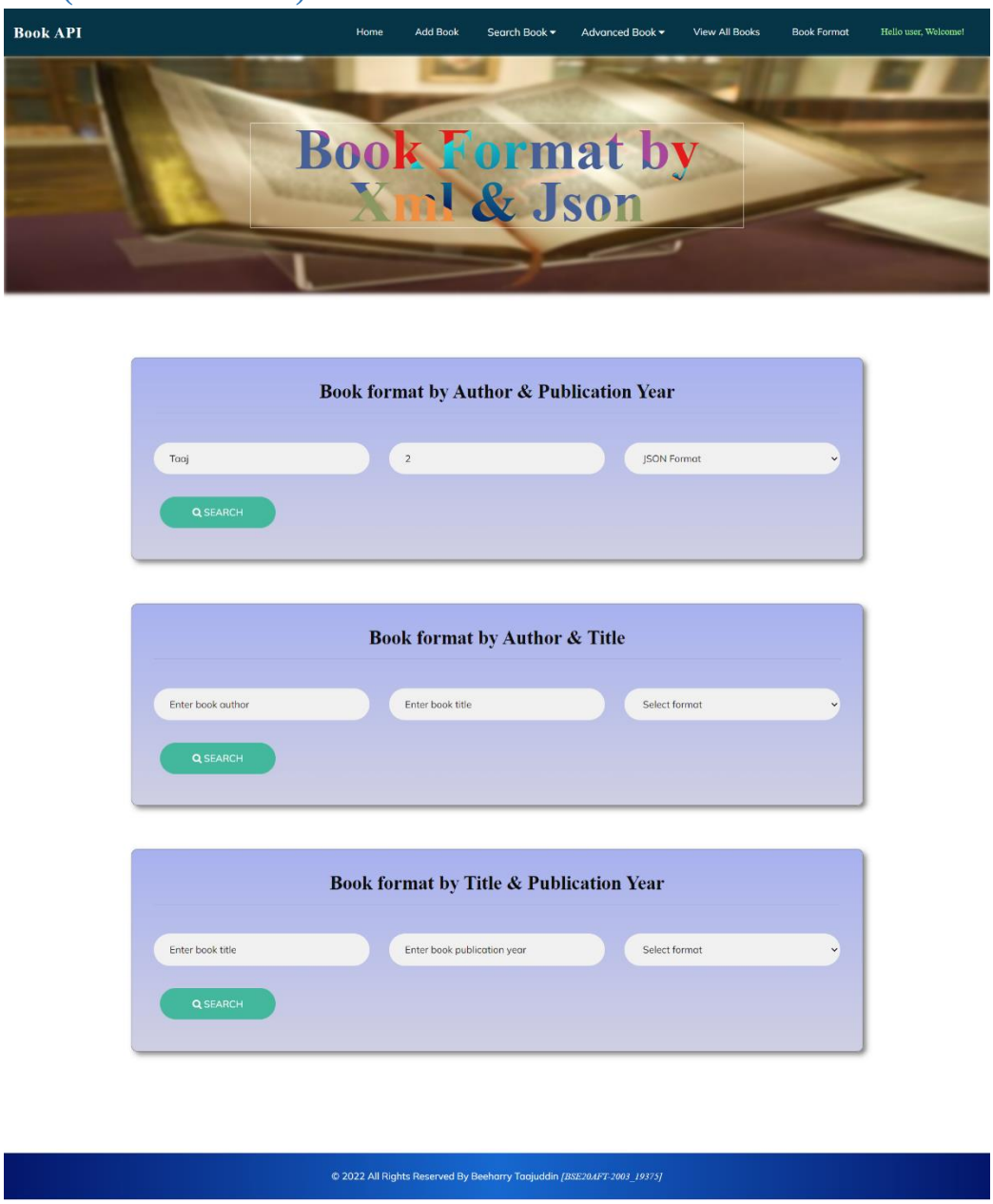


Figure 16 - Book format (bookformat.html)

Figure 16 shows the book format page utilized by programmers. A programmer can choose the response type he desires.

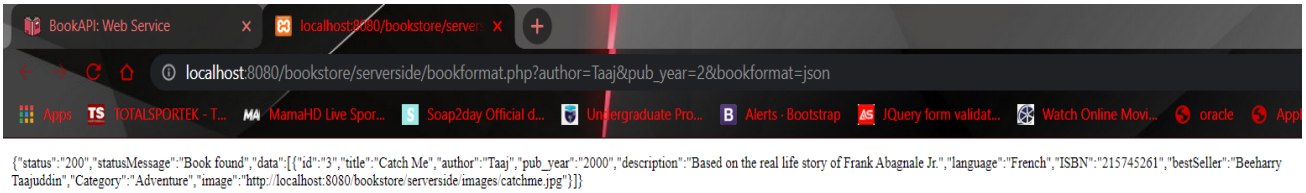


Figure 17 - Json book format

Figure 17 shows the json book format.

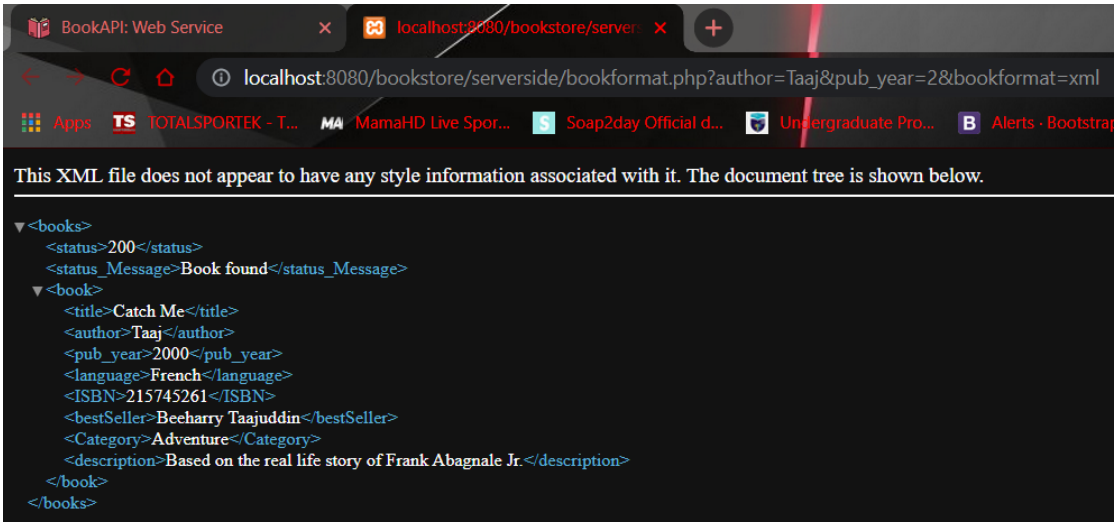


Figure 18 - Xml book format

Figure 18 shows the xml book format.

Implementation

bookFunctions.php (Server Side)

```
41  function getAllDetails($title){
42      //connection object $conn holds the connection to the Database
43      $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
44      $title = strtolower($title);
45
46      //TODO
47      $query = "select * from books where title LIKE '%$title%'";
48      $result = mysqli_query($conn, $query);
49      $book_details = null;
50
51      if($result){
52          while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
53              $data[]=$row;
54          }
55      return $data;
56      }
57
58  }
59
60  function getBookByAuthor($author){
61      //connection object $conn holds the connection to the Database
62      $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
63      $author = strtolower($author);
64
65      //TODO
66      $query = "select * from books where author LIKE '%$author%'";
67      $result = mysqli_query($conn, $query);
68      $book_details = null;
69
70      if($result){
71          $data=array();
72          while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
73              $data[]=$row;
74          }
75      return $data;
76      }
77
78  }
```

Figure 19 - Get book by title, author (bookFunctions.php)

Figure 19 depicts the implemented code for server side (*bookFunctions.php*). At *line 44*, the **$title** parameter is passed to the function and is concatenated with the sql **$query** (*line 47*). For the select statement **LIKE '%$title%'** has been utilized so that user can search with starting characters. The title is being used to obtain information about the books and an array is used to hold all of the output.

At *line 63*, **$author** is passed to the function and is concatenated with the sql **$query** (*line 66*). For the select statement **LIKE '%$author%'** has been utilized so that user can search with starting characters.

The name of author is being used to obtain information about the books and an array is used to hold all of the output.

```
80  function getBookByYear($year){
81      //connection object $conn holds the connection to the Database
82      $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
83      $year = strtolower($year);
84
85      //TODO
86      $query = "select * from books where pub_year LIKE '%$year%'";
87      $result = mysqli_query($conn, $query);
88      $book_details = null;
89
90      if($result){
91          $data=array();
92          while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
93              $data[]=$row;
94          }
95      return $data;
96      }
97
98  }
99
100 function getBookByCategory($category){
101     //connection object $conn holds the connection to the Database
102     $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
103     $category = strtolower($category);
104
105     //TODO
106     $query = "select * from books where Category LIKE '%$category%'";
107     $result = mysqli_query($conn, $query);
108     $book_details = null;
109
110     if($result){
111         $data=array();
112         while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
113             $data[]=$row;
114         }
115     return $data;
116     }
117
118 }
```

Figure 20 - Get book by year, category (bookFunctions.php)

Figure 20 depicts the implemented code for server side (*bookFunctions.php*). At *line 83*, the **$year** parameter is passed to the function and is concatenated with the sql **$query** (*line 86*). For the select statement **LIKE '%$year%'** has been utilized so that user can search with starting characters. The publication year is being used to obtain information about the books and an array is used to hold all of the output.

At *line 103*, **$category** is passed to the function and is concatenated with the sql **$query** (*line 106*). For the select statement **LIKE '%$category %'** has been utilized so that user can search with starting characters. Category is being used to obtain information about the books and an array is used to hold all of the output.

```php
function getBookByAuthorCat($author,$category){
    //connection object $conn holds the connection to the Database
    $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());

    //TODO
    $query = "select * from books where author LIKE '%$author%' AND Category LIKE '%$category%'";
    $result = mysqli_query($conn, $query);
    $book_details = null;

    if($result){
        $data=array();
        while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
            $data[]=$row;
        }
    return $data;
    }

}

function getBookByAuthorYear($author,$year){
    //connection object $conn holds the connection to the Database
    $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());

    //TODO
    $query = "select * from books where author LIKE '%$author%' AND pub_year LIKE '%$year%'";
    $result = mysqli_query($conn, $query);
    $book_details = null;

    if($result){
        $data=array();
        while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
            $data[]=$row;
        }
    return $data;
    }

}
```

Figure 21 - Get book by author & category, author & year (bookFunctions.php)

Figure 21 depicts the implemented code for server side (*bookFunctions.php*). **$author** and **$category** are passed to the function and concatenated with the sql **$query** (*line 125*). For the select statement **LIKE '%$author%'** and **LIKE '%$category%'** have been utilized so that user can search with starting characters. Author name and book category are being used to obtain information about the books and an array is used to hold all of the output.

$author and **$year** are passed to the function and concatenated with the sql **$query** (*line 144*). For the select statement **LIKE '%$author%'** and **LIKE '%$year%'** have been utilized so that user can search with starting characters. Author name and publication year are being used to obtain information about the books and an array is used to hold all of the output.

```php
function getAllBookByData(){
    //connection object $conn holds the connection to the Database
    $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());

    //TODO
    $query = "select * from books";
    $result = mysqli_query($conn, $query);
    $book_details = null;

    if($result){
        $data=array();
        while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
            $data[]=$row;
        }
    return $data;
    }
}
```

Figure 22 - Get all books (bookFunctions.php)

The function to get all the books from the database is shown in figure 22, and the output is stored in an array.

```php
function addBook($title, $author, $pub_year, $description, $language, $ISBN, $bestSeller, $Category){
    //connection object $conn holds the connection to the Database
    $dbconn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: '.mysqli_connect_error());
    $image="http://localhost:8080/bookstore/serverside/images/bookcover.jpg";

    //TODO
    $query = "INSERT INTO books (title, author, pub_year, description, language, ISBN, bestSeller, Category, image)VALUES ('".$title."', '".$author."', '".$
        pub_year."', '".$description."', '".$language."', '".$ISBN."', '".$bestSeller."', '".$Category."', '".$image."')";

    if (mysqli_query($dbconn, $query)) {
        echo "Book is successfully added.";
    } else {
        echo "Error: " . $query . "" . mysqli_error($dbconn);
    }
    $dbconn->close();
}
```

Figure 23 - Add book (bookFunctions.php)

Figure 23 depicts the add book function where books are inserted into the database. The code for the image has been hardcoded before the insert statement.

```
function getBookByAYFormat($author,$year){

		//connection object $conn holds the connection to the Database

		$conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' .
mysqli_connect_error());



		//TODO

		$query = "select * from books where author LIKE '%$author%' AND pub_year LIKE '%$year%'";

		$result = mysqli_query($conn, $query);



		$book_details = null;



		if($result){

			$data=array();

			while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){

				$data[]=$row;

			}

		return $data;

		}

}



function getBookByATFormat($author,$title){

		//connection object $conn holds the connection to the Database

		$conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' .
mysqli_connect_error());



		//TODO

		$query = "select * from books where author LIKE '%$author%' AND title LIKE '%$title%'";

		$result = mysqli_query($conn, $query);



		$book_details = null;



		if($result){

			$data=array();

			while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){

				$data[]=$row;

			}

		return $data;

		}

}
```

Figure 24 - Get book format (bookFunctions.php(1))

```
function getBookByTYFormat($title,$year){

		//connection object $conn holds the connection to the Database

		$conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' .
mysqli_connect_error());



		//TODO

		$query = "select * from books where title LIKE '%$title%' AND pub_year LIKE '%$year%'";

		$result = mysqli_query($conn, $query);



		$book_details = null;



		if($result){

			$data=array();

			while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){

				$data[]=$row;

			}

		return $data;

		}

}
```

Figure 25 - Get book format (bookFunctions.php(2))

The function is seen in Figures 24 and 25 while a user is looking for books.

index.php (Server Side)

```
<?php
header("Content-Type:application/json");
include("bookFunctions.php");
if(!empty($_GET['title'])){
                $books = getAllDetails($_GET['title']);
                if(empty($books)){
                  deliver_response('200', 'No Book Found', null);
                }else{
                  deliver_response('200', 'Book found', $books);
                }
}else if(!empty($_GET['author']) && !empty($_GET['Category'])){
   $books = getBookByAuthorCat($_GET['author'],$_GET['Category']);
   if(empty($books)){
     deliver_response('200', 'No Book Found', null);
   }else{
     deliver_response('200', 'Book found', $books);
   }
}else if(!empty($_GET['author']) && !empty($_GET['pub_year'])){
   $books = getBookByAuthorYear($_GET['author'],$_GET['pub_year']);
   if(empty($books)){
     deliver_response('200', 'No Book Found', null);
   }else{
     deliver_response('200', 'Book found', $books);
   }
}else if(!empty($_GET['author'])){
                $books = getBookByAuthor($_GET['author']);
                if(empty($books)){
                  deliver_response('200', 'No Book Found', null);
                }else{
                  deliver_response('200', 'Book Found', $books);
                }
}else if(!empty($_GET['pub_year'])){
                $books = getBookByYear($_GET['pub_year']);
                if(empty($books)){
                  deliver_response('200', 'No Book Found', null);
                }else{
                  deliver_response('200', 'Book found', $books);
                }
}else if(!empty($_GET['Category'])){
                $books = getBookByCategory($_GET['Category']);
                if(empty($books)){
```

Figure 26 - index.php (1)

```
                    deliver_response('200', 'No Book Found', null);
                }else{
                    deliver_response('200', 'Book found', $books);
                }
}else if(!empty($_GET['data'])){
                $books = getAllBookByData($_GET['data']);
                if(empty($books)){
                    deliver_response('200', 'No Book Found', null);
                }else{
                    deliver_response('200', 'Book found', $books);
                }
}else{
                deliver_response('400', 'Bad Request', null);
}

function deliver_response($status, $statusMessage, $data){
 header("HTTP/1.1 $status $statusMessage");
 $response['status']=$status;
 $response['status_message']=$statusMessage;
 $response['data']=$data;

 $json_response=json_encode($response, JSON_UNESCAPED_SLASHES);
 echo $json_response;
}

?>
```

Figure 27 - index.php (2)

Figure 26 and 27 show the index.php server side.

Add Book

```php
<?php

include("bookFunctions.php");

$title = null;
$author = null;
$pub_year = null;
$description = null;
$language = null;
$ISBN = null;
$bestSeller = null;
$Category = null;


if(!empty($_GET['title']) && !empty($_GET['author']) && !empty($_GET['pub_year']) && !empty($_GET['description']) && !empty($_GET['language']) &&
!empty($_GET['ISBN']) && !empty($_GET['bestSeller']) && !empty($_GET['Category']))
{

    $title = $_GET['title'];
    $author = $_GET['author'];
    $pub_year = $_GET['pub_year'];
    $description = $_GET['description'];
    $language = $_GET['language'];
    $ISBN = $_GET['ISBN'];
    $bestSeller = $_GET['bestSeller'];
    $Category = $_GET['Category'];

    if($title != null && $author != null && $pub_year != null && $description != null && $language != null && $ISBN != null && $bestSeller != null && $Category
        != null)
    {
        addBook($title, $author, $pub_year, $description, $language, $ISBN, $bestSeller, $Category);
    }

}
else{
    echo "All text fields are required!!";
}


?>
```

Figure 28 - Add book Server Side (addbook.php)

Figure 28 shows the **$_GET** is utilized to get all of the book's information from the url below figure 30 and each value is saved in a variable. The if statement is used to determine whether or not a variable is null. If the variables holding the book's information are not null, they are passed to the function **addBook()**.

```html
<div class="backg">
  <div class="row">
    <div class="col-md-6"><input type="text" id="btitle" placeholder="Enter book title" /></div>
    <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
    <div class="col-md-6"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
    <div class="col-md-6"><input type="text" id="bdes" placeholder="Enter book description" /></div>
    <div class="col-md-6"><input type="text" id="blang" placeholder="Enter book language" /></div>
    <div class="col-md-6"><input type="text" id="bisbn" placeholder="Enter book isbn" /></div>
    <div class="col-md-6"><input type="text" id="bbseller" placeholder="Enter book best seller" /></div>
    <div class="col-md-6"><input type="text" id="bcat" placeholder="Enter book category" /></div>
  </div>
  <div class="btn-box">
    <button onclick="Submit()" >SUBMIT</button>
  </div>
    <p id="msg"></p>
</div>
```

Figure 29 - Add book Client Side (addbook.html)

Figure 29 shows the add book form and each input tag has its own id and values will be obtained by the user that they have entered using the id. An **onclick="Submit()"** is used to call the function submit as shown in figure 30 below.

```
<script>
  function Submit() {
    var title = document.getElementById('btitle').value;
    var author = document.getElementById('bauthor').value;
    var pub_year = document.getElementById('byear').value;
    var description = document.getElementById('bdes').value;
    var language = document.getElementById('blang').value;
    var ISBN = document.getElementById('bisbn').value;
    var bestSeller = document.getElementById('bbseller').value;
    var Category = document.getElementById('bcat').value;

    if(title == "" || author =="" || pub_year =="" || description =="" || language =="" || ISBN =="" || bestSeller =="" || Category ==""){
      msg.textContent = "Please fill all the textboxes!"
      msg.style.color = "darkred"
      return false;

    }else {
      var url = "http://localhost:8080/bookstore/serverside/addbook.php?title="+title+"&author="+author+"&pub_year="+pub_year+"&description="+description+"
        &language="+language+"&ISBN="+ISBN+"&bestSeller="+bestSeller+"&Category="+Category;

      var xhttp = new XMLHttpRequest();

      xhttp.open("POST", url, true);
      xhttp.send(title, author, pub_year, description, language, ISBN, bestSeller, Category);
      //window.location.href = "addbook.html";

      msg.textContent = "Book is successfully added."
      msg.style.color = "green"

      var title = "";
      var author = "";
      var pub_year = "";
      var description = "";
      var language = "";
      var ISBN = "";
      var bestSeller = "";
      var Category = "";
    }
  }
  </script>
```

Figure 30 - Add book script Client Side (addbook.html)

The *Submit()* function is shown id figure 30. At first, the variables are declared and the id is used to obtain all of the values inserted. Each variable stores all of the data which is then passed to the url. An if statement id used to check for empty textboxes. In case the textboxes are empty, a message will display in *red* that *Please fill all the textboxes* else adding a book a message will appear in *green* that *Book is successfully added*.

Search Book

By title

```
            <b>Search book by title</b>
          </h3><hr style="margin-bottom: 45px;" />
          <div>
            <input type="text" id="btitle" placeholder="Enter book title" />
          </div>
            <div class="btn-box">
              <button onclick="SearchByTitle()" >
                <i class="fa fa-search" aria-hidden="true"></i> SEARCH
              </button>
          </div><br/><div id="displayBtitle"></div>
```

Figure 31 - Search book by title (searchbooktitle.html)

In figure 31, it shows the code for searching for a book by title. The input element is utilized and an id *btitle* was given. The function id called when the button is clicked. The **id=“displayBtitle”** is used to display the book results.

```
<script>
  function SearchByTitle(){
    var title = document.getElementById("btitle").value;
    var url="http://localhost:8080/bookstore/serverside/index.php?title="+title;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        var book = JSON.parse(this.responseText);

        var result="";
        result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

        for(var i = 0; i < book.data.length; i++){
          result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
          +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
          +"<b>Author: </b>"+book.data[i].author
          +"<br><b>Year Published: </b>"+book.data[i].pub_year
          +"<br><b>Language: </b>"+book.data[i].language
          +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
          +"<br><b>ISBN: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].ISBN+"</i>"
          +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
          +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
          +"</div>";
        }
        document.getElementById("displayBtitle").innerHTML = result;

      }else{
        document.getElementById("displayBtitle").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
      }
    };
    xmlhttp.open("GET", url, true);
    xmlhttp.send();
  }

</script>
```

Figure 32 - Search book by title script (searchbooktitle.html)

The code that is executed when the onclick button is clicked is shown in figure 32. The *title* variable is declared which is then concatenated with the url. **New XMLHttpRequest()** is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayBtitle").innerHTML**. In case the provided id does not match, a message will appear (*line 235*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

By author

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Search book by author</b>
</h3><hr style="margin-bottom: 45px;" />
<div>
  <input type="text" id="bauthor" placeholder="Enter book author" />
</div>
  <div class="btn-box">
    <button onclick="SearchByAuthor()" >
      <i class="fa fa-search" aria-hidden="true"></i> SEARCH
    </button>
  </div><br/><div id="displayBauthor"></div>
```

Figure 33 - Search book by author (searchbookauthor.html)

In figure 33, it shows the code for searching for a book by author. The input element is utilized and an id *bauthor* was given. The function id called when the button is clicked. The **id="displayBauthor"** is used to display the book results.

```
<script>
  function SearchByAuthor(){
    var author = document.getElementById("bauthor").value;
    var url="http://localhost:8080/bookstore/serverside/index.php?author="+author;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        var book = JSON.parse(this.responseText);

        var result="";
        result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

        for(var i = 0; i < book.data.length; i++){
          result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
          +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
          +"<b>Author: </b>"+book.data[i].author
          +"<br><b>Year Published: </b>"+book.data[i].pub_year
          +"<br><b>Language: </b>"+book.data[i].language
          +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
          +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
          +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
          +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
          +"</div>";
        }
        document.getElementById("displayBauthor").innerHTML = result;

      }else{
        document.getElementById("displayBauthor").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
      }
    };
    xmlhttp.open("GET", url, true);
    xmlhttp.send();
  }

</script>
```

Figure 34 - Search book by author script (searchbookauthor.html)

The code that is executed when the onclick button is clicked is shown in figure 34. The *author* variable is declared which is then concatenated with the url. **New XMLHttpRequest()** is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayBauthor").innerHTML**. In case the provided id does not match, a message will appear (*line 235*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

By publication year

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Search book by publication year</b>
</h3><hr style="margin-bottom: 45px;" />
<div>
  <input type="text" id="byear" placeholder="Enter book publication year" />
</div>
  <div class="btn-box">
    <button onclick="SearchByYear()" >
      <i class="fa fa-search" aria-hidden="true"></i> SEARCH
    </button>
  </div><br/><div id="displayByear"></div>
```

Figure 35 - Search book by pub year (searchbookyear.html)

In figure 35, it shows the code for searching for a book by publication year. The input element is utilized and an id *byear* was given. The function id called when the button is clicked. The **id="displayByear"** is used to display the book results.

```
<script>
  function SearchByYear(){
    var pub_year = document.getElementById("byear").value;
    var url="http://localhost:8080/bookstore/serverside/index.php?pub_year="+pub_year;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        var book = JSON.parse(this.responseText);

        var result="";
        result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

        for(var i = 0; i < book.data.length; i++){
          result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
          +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
          +"<b>Author: </b>"+book.data[i].author
          +"<br><b>Year Published: </b>"+book.data[i].pub_year
          +"<br><b>Language: </b>"+book.data[i].language
          +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
          +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
          +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
          +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
          +"</div>";
        }
        document.getElementById("displayByear").innerHTML = result;

      }else{
        document.getElementById("displayByear").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
      }
    };
    xmlhttp.open("GET", url, true);
    xmlhttp.send();
  }

</script>
```

Figure 36 - Search book by pub year script (searchbookyear.html)

The code that is executed when the onclick button is clicked is shown in figure 36. The *pub_year* variable is declared which is then concatenated with the url. **New XMLHttpRequest()** is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayByear").innerHTML**. In case the provided id does not match, a message will appear (*line 235*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

By category

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Search book by category</b>
</h3><hr style="margin-bottom: 45px;" />
<div>
  <input type="text" id="bcat" placeholder="Enter book category" />
</div>
  <div class="btn-box">
    <button onclick="SearchByCat()" >
      <i class="fa fa-search" aria-hidden="true"></i> SEARCH
    </button>
  </div><br/><div id="displayBcat"></div>
```

Figure 37 - Search book by category (searchbookcat.html)

In figure 37, it shows the code for searching for a book by category. The input element is utilized and an id *bcat* was given. The function id called when the button is clicked. The **id="displayBcat"** is used to display the book results.

```
<script>
  function SearchByCat(){
    var Category = document.getElementById("bcat").value;
    var url="http://localhost:8080/bookstore/serverside/index.php?Category="+Category;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        var book = JSON.parse(this.responseText);

        var result="";
        result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

        for(var i = 0; i < book.data.length; i++){
          result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
          +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
          +"<b>Author: </b>"+book.data[i].author
          +"<br><b>Year Published: </b>"+book.data[i].pub_year
          +"<br><b>Language: </b>"+book.data[i].language
          +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
          +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
          +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
          +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
          +"</div>";
        }
        document.getElementById("displayBcat").innerHTML = result;

      }else{
        document.getElementById("displayBcat").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
      }
    };
    xmlhttp.open("GET", url, true);
    xmlhttp.send();
  }

</script>
```

Figure 38 - Search book by category script (searchbookcat.html)

The code that is executed when the onclick button is clicked is shown in figure 38. The *Category* variable is declared which is then concatenated with the url. **New XMLHttpRequest()** is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayBcat").innerHTML**. In case the provided id does not match, a message will appear (*line 235*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

Advanced Book

By author & category

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Search book by author & category</b>
</h3><hr style="margin-bottom: 45px;" />
<div class="row">
  <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
  <div class="col-md-6"><input type="text" id="bcat" placeholder="Enter book category" /></div>
</div>
  <div class="btn-box">
    <button onclick="SearchByAuthorCat()" >
      <i class="fa fa-search" aria-hidden="true"></i> SEARCH
    </button>
  </div><br/><p id="msg"></p><div id="displayBAuCat"></div>
```

Figure 39 - Search book by author & category (advancedsearchAC.html)

In figure 39, it shows the code for searching for a book by author and category. The input element is utilized and an id *bauthor* and *bcat* were given. The function id called when the button is clicked. The **id="displayBAuCat"** is used to display the book results.

```
<script>
  function SearchByAuthorCat(){
    var author = document.getElementById("bauthor").value;
    var Category = document.getElementById("bcat").value;
    if(author == "" || Category ==""){
      msg.textContent = "Author and Category are required!"
      msg.style.color = "darkred"
      return false;

    }else{

      var url="http://localhost:8080/bookstore/serverside/index.php?author="+author+"&Category="+Category;

      var xmlhttp = new XMLHttpRequest();
      xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
          var book = JSON.parse(this.responseText);

          var result="";
          result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

          for(var i = 0; i < book.data.length; i++){
            result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
            +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
            +"<b>Author: </b>"+book.data[i].author
            +"<br><b>Year Published: </b>"+book.data[i].pub_year
            +"<br><b>Language: </b>"+book.data[i].language
            +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
            +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
            +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
            +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
            +"</div>";
          }
          document.getElementById("displayBAuCat").innerHTML = result;

        }else{
          document.getElementById("displayBAuCat").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
        }
      };
      xmlhttp.open("GET", url, true);
      xmlhttp.send();
    }
```

Figure 40 - Search book by author & category script (advancedsearchAC.html)

The code that is executed when the onclick button is clicked is shown in figure 40. The *author* and *Category* variable are declared which is then concatenated with the url. Moreover, an if statement has been utilized to check whether the textboxes are null. If the textboxes are null, a message will display in *red* that *Author and Category are required*.

New XMLHttpRequest() is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayBAuCat").innerHTML**. In case the provided id does not match, a message will appear (*line 243*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

By author & year

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Search book by author & publication year</b>
</h3><hr style="margin-bottom: 45px;" />
<div class="row">
  <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
  <div class="col-md-6"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
</div>
  <div class="btn-box">
    <button onclick="SearchByAuthorYear()" >
      <i class="fa fa-search" aria-hidden="true"></i> SEARCH
    </button>
</div><br/><p id="msg"></p><div id="displayBAuYear"></div>
```

Figure 41 - Search book by author & pub year (advancedsearchAY.html)

In figure 41, it shows the code for searching for a book by author and publication year. The input element is utilized and an id *bauthor* and *byear* were given. The function id called when the button is clicked. The **id="displayBAuYear"** is used to display the book results.

```
<script>
  function SearchByAuthorYear(){
    var author = document.getElementById("bauthor").value;
    var pub_year = document.getElementById("byear").value;
    if(author == "" || pub_year ==""){
      msg.textContent = "Author and Publication Year are required!"
      msg.style.color = "darkred"
      return false;

    }else{

      var url="http://localhost:8080/bookstore/serverside/index.php?author="+author+"&pub_year="+pub_year;

      var xmlhttp = new XMLHttpRequest();
      xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
          var book = JSON.parse(this.responseText);

          var result="";
          result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";

          for(var i = 0; i < book.data.length; i++){
            result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
            +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
            +"<b>Author: </b>"+book.data[i].author
            +"<br><b>Year Published: </b>"+book.data[i].pub_year
            +"<br><b>Language: </b>"+book.data[i].language
            +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
            +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
            +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
            +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
            +"</div>";
          }
          document.getElementById("displayBAuYear").innerHTML = result;

        }else{
          document.getElementById("displayBAuYear").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
        }
      };
      xmlhttp.open("GET", url, true);
      xmlhttp.send();
    }
```

Figure 42 - Search book by author & pub year script (advancedsearchAY.html)

The code that is executed when the onclick button is clicked is shown in figure 42. The *author* and *pub_year* variable are declared which is then concatenated with the url. Moreover, an if statement has been utilized to check whether the textboxes are null. If the textboxes are null, a message will display in *red* that *Author and Publication Year are required*.

New XMLHttpRequest() is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayBAuYear").innerHTML**. In case the provided id does not match, a message will appear (*line 243*). The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

View All Book

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>View all books</b>
</h3><hr style="margin-bottom: 45px;" />
<br/><div id="displayAllBooks"></div>
```

Figure 43 - View all books (viewallbooks.html)

In figure 43, it shows the code for displaying all the books which are in the database by the **id="displayAllBooks"**.

```
<script>
    var url="http://localhost:8080/bookstore/serverside/index.php?data[]";

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
        var book = JSON.parse(this.responseText);

        var result="";
        result+="<br/> <h3 style='font-family: sans-serif;'>"+"Total number of books: "+book.data.length+"</h3><br/>";

        for(var i = 0; i < book.data.length; i++){
          result+="<div class='card1'>"+"<img  class='card-img' src='"+book.data[i].image+ "'style='width:100%;height:340px;'"+"' alt='No Image'"+"/>"
          +"<br><br><h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"
          +"<b>Author: </b>"+book.data[i].author
          +"<br><b>Year Published: </b>"+book.data[i].pub_year
          +"<br><b>Language: </b>"+book.data[i].language
          +"<br><b>Category: </b>"+"<i>"+book.data[i].Category+"</i>"
          +"<br><b>ISBN: </b>"+"<i>"+book.data[i].ISBN+"</i>"
          +"<br><b>Description: </b>"+"<i style='font-family: sans-serif;'>"+book.data[i].description+"</i>"
          +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "+"<i>"+book.data[i].bestSeller+"</i>"+"</b>"
          +"</div>";
        }
        document.getElementById("displayAllBooks").innerHTML = result;

    };
    xmlhttp.open("GET", url, true);
    xmlhttp.send();

</script>
```

Figure 44 - View all books script (viewallbooks.html)

Figure 44 shows the url for displaying all the books. New **XMLHttpRequest()** is used to assign the variable **xmlhttp**. The XMLHttpRequest's status is stored in the *readyState* attribute. if ***readyState == 4 && readyState == 200***, the response is parsed into JSON format and stored in variable book using **JSON.parse(this.responseText)**.

To loop into the object and document, the for loop is utilized. The result is shown by using **getElementById("displayAllBooks").innerHTML**. The **xmlhttp.open("Get", url, true)** is utilized to provide the kind of request and to send a request to the server, **xmlhttp.send()** is used.

Book Format

```
<?php
include("bookFunctions.php");
$bookformat="";

if (!empty($_GET['bookformat'])) {
   $bookformat = $_GET['bookformat'];
}

if(!empty($_GET['author']) && !empty($_GET['pub_year'])){
   $books = getBookByAYFormat($_GET['author'],$_GET['pub_year']);
   if($bookformat == 'json'){
       if(empty($books)){
     deliver_responseJson('200', 'No Book Found', null);
   }else{
     deliver_responseJson('200', 'Book found', $books);
   }
   }
  else if($bookformat == 'xml'){
       if(empty($books)){
     deliver_responseXML('200', 'No Book Found', null);
   }else{
     deliver_responseXML('200', 'Book found', $books);
   }
  }
}else if(!empty($_GET['author']) && !empty($_GET['title'])){
   $books = getBookByATFormat($_GET['author'],$_GET['title']);
   if($bookformat == 'json'){
       if(empty($books)){
     deliver_responseJson('200', 'No Book Found', null);
   }else{
     deliver_responseJson('200', 'Book found', $books);
   }
   }
  else if($bookformat == 'xml'){
       if(empty($books)){
     deliver_responseXML('200', 'No Book Found', null);
   }else{
     deliver_responseXML('200', 'Book found', $books);
   }
  }
}else if(!empty($_GET['title']) && !empty($_GET['pub_year'])){
```

Figure 45 - bookformat.php (1)

```
  $books = getBookByTYFormat($_GET['title'],$_GET['pub_year']);
  if($bookformat == 'json'){
      if(empty($books)){
    deliver_responseJson('200', 'No Book Found', null);
  }else{
    deliver_responseJson('200', 'Book found', $books);
  }
  }
 else if($bookformat == 'xml'){
      if(empty($books)){
    deliver_responseXML('200', 'No Book Found', null);
  }else{
    deliver_responseXML('200', 'Book found', $books);
  }
 }
}

function deliver_responseXML($Status, $status_message, $data){
  header("Content-Type:application/xml");
  header("HTTP/1.1 $Status $status_message");

  $xml = new SimpleXMLElement('<books/>');

  if ($data != NULL){
    $xml->addChild('status', $Status);
    $xml->addChild('status_Message', $status_message);

  for($i = 0; $i < count($data); $i++){
    $book = $xml->addChild('book');
    $book->addChild('title', $data[$i]['title']);
    $book->addChild('author', $data[$i]['author']);
    $book->addChild('pub_year', $data[$i]['pub_year']);
    $book->addChild('language', $data[$i]['language']);
    $book->addChild('ISBN', $data[$i]['ISBN']);
    $book->addChild('bestSeller', $data[$i]['bestSeller']);
    $book->addChild('Category', $data[$i]['Category']);
    $book->addChild('description', $data[$i]['description']);
   }
  }
  else {
    $xml->addChild('status', $Status);
```

Figure 46 - bookformat.php (2)

```
        $xml->addChild('Status_Message', $status_message);
    }
        echo $xml->asXML();
}



function deliver_responseJson( $status, $statusMessage, $data){
    header("Content-Type:application/json");
    header("HTTP/1.1 $status $statusMessage");
    $response['status'] = $status;
    $response['statusMessage'] = $statusMessage;
    $response['data'] = $data;

    $json_response = json_encode($response, JSON_UNESCAPED_SLASHES);
    echo $json_response;
}
?>
```

Figure 47 - bookformat.php (3)

The format *author and year*, *author and title*, *title and year* are obtained from url using **$_GET** as shown in figure 45 and 46. The function **deliver_responseJson**, as shown in figure 47, is used to transform the data to Json, if Json is chosen as the format. In the other hand, the function deliver_reponseXML, as shown in figure 46, will be invoked to convert the data if user has chosen the xml format. A deliver_responseJson is a function that convert data to json format and a deliver_responseXML is a function that convert data to xml format.

Search by author & publication year

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Book format by Author & Publication Year</b>
</h3><hr style="margin-bottom: 45px;" />
<div class="row">
  <div class="col-md-4"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
  <div class="col-md-4"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
  <div class="col-md-4">
    <select name="format" class="txtdrop" id="AYformat">
      <option class="txtdrop1" value="">Select format</option>
      <option class="txtdrop1" value="json">JSON Format</option>
      <option class="txtdrop1" value="xml">XML Format</option>
    </select>
  </div>
</div>
<div class="btn-box">
  <button onclick="SearchByAuthorYearFormat1()" >
    <i class="fa fa-search" aria-hidden="true"></i> SEARCH
  </button>
</div><p id="msg"></p>
```

Figure 48 - Book format author & pub year (bookformat.html)

In figure 48, it shows the code for searching for a book by author and publication year. The input element is utilized and an id *bauthor* and *byear* were given. Furthermore, a dropdown is used and an id *AYformat* is given for the book format. The function id called when the button is clicked. The **id="msg"** is used to display the book results.

```
<script>
  function SearchByAuthorYearFormat1(){
    var author = document.getElementById("bauthor").value;
    var pub_year = document.getElementById("byear").value;
    var bookformat = document.getElementById("AYformat").value;
    if(author == "" || pub_year =="" || bookformat ==""){
      msg.textContent = "Please fill all the textboxes!"
      msg.style.color = "darkred"
      return false;

    }else{

      if (bookformat=='json'){
        window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&pub_year="+pub_year+"&bookformat="+bookformat);
      }else if(bookformat=='xml'){
        window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&pub_year="+pub_year+"&bookformat="+bookformat);
      }
    }
  }
```

Figure 49 - Book format author & pub year script (bookformat.html)

The function that is done when the search button is pressed is shown in figure 49. The *author* and *pub_year* variable are declared which is then concatenated with the url. An if statement is utilized to check whether the textboxes are empty. If they are empty, a message will be displayed in *red* that *Please fill all the textboxes*. Else another if statement is utilized to check if the book format is json or xml and if bookformat==json, the response will be displayed in json format else if bookformat==xml, the response will be displayed in xml format.

Search by author & title

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Book format by Author & Title</b>
</h3><hr style="margin-bottom: 45px;" />
<div class="row">
  <div class="col-md-4"><input type="text" id="bauthor1" placeholder="Enter book author" /></div>
  <div class="col-md-4"><input type="text" id="btitle" placeholder="Enter book title" /></div>
  <div class="col-md-4">
    <select name="format" class="txtdrop" id="ATformat">
      <option class="txtdrop1" value="">Select format</option>
      <option class="txtdrop1" value="json">JSON Format</option>
      <option class="txtdrop1" value="xml">XML Format</option>
    </select>
  </div>
</div>
<div class="btn-box">
  <button onclick="SearchByAuthorTitleFormat()" >
    <i class="fa fa-search" aria-hidden="true"></i> SEARCH
  </button>
</div><p id="msg1"></p>
```

Figure 50 - Book format author & title (bookformat.html)

In figure 50, it shows the code for searching for a book by author and title. The input element is utilized and an id *bauthor* and *btitle* were given. Furthermore, a dropdown is used and an id *AYformat* is given for the book format. The function id called when the button is clicked. The **id="msg1"** is used to display the book results.

```
function SearchByAuthorTitleFormat(){
  var author = document.getElementById("bauthor1").value;
  var title = document.getElementById("btitle").value;
  var bookformat = document.getElementById("ATformat").value;
  if(author == "" || title =="" || bookformat ==""){
    msg1.textContent = "Please fill all the textboxes!"
    msg1.style.color = "darkred"
    return false;

  }else{

    if (bookformat=='json'){
      window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&title="+title+"&bookformat="+bookformat);
    }else if(bookformat=='xml'){
      window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&title="+title+"&bookformat="+bookformat);
    }
  }
}
```

Figure 51 - Book format author & title script (bookformat.html)

The function that is done when the search button is pressed is shown in figure 51. The *author* and *title* variable are declared which is then concatenated with the url. An if statement is utilized to check

whether the textboxes are empty. If they are empty, a message will be displayed in *red* that *Please fill all the textboxes*. Else another if statement is utilized to check if the book format is json or xml and if bookformat==json, the response will be displayed in json format else if bookformat==xml, the response will be displayed in xml format.

Search by title & publication year

```
<h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
  <b>Book format by Title & Publication Year</b>
</h3><hr style="margin-bottom: 45px;" />
<div class="row">
  <div class="col-md-4"><input type="text" id="btitle1" placeholder="Enter book title" /></div>
  <div class="col-md-4"><input type="text" id="byear1" placeholder="Enter book publication year" /></div>
  <div class="col-md-4">
    <select name="format" class="txtdrop" id="TYformat">
      <option class="txtdrop1" value="">Select format</option>
      <option class="txtdrop1" value="json">JSON Format</option>
      <option class="txtdrop1" value="xml">XML Format</option>
    </select>
  </div>
</div>
<div class="btn-box">
  <button onclick="SearchByTitleYearFormat()" >
    <i class="fa fa-search" aria-hidden="true"></i> SEARCH
  </button>
</div><p id="msg2"></p>
```

Figure 52 - Book format title & pub year (bookformat.html)

In figure 52, it shows the code for searching for a book by title and publication year. The input element is utilized and an id *btitle1* and *byear1* were given. Furthermore, a dropdown is used and an id *TYformat* is given for the book format. The function id called when the button is clicked. The **id="msg2"** is used to display the book results.

```
function SearchByTitleYearFormat(){
  var title = document.getElementById("btitle1").value;
  var pub_year = document.getElementById("byear1").value;
  var bookformat = document.getElementById("TYformat").value;
  if(title == "" || pub_year =="" || bookformat ==""){
    msg2.textContent = "Please fill all the textboxes!"
    msg2.style.color = "darkred"
    return false;

  }else{

    if (bookformat=='json'){
      window.open("http://localhost:8080/bookstore/serverside/bookformat.php?title="+title+"&pub_year="+pub_year+"&bookformat="+bookformat);
    }else if(bookformat=='xml'){
      window.open("http://localhost:8080/bookstore/serverside/bookformat.php?title="+title+"&pub_year="+pub_year+"&bookformat="+bookformat);
    }
  }
}
```

Figure 53 - Book format author & title script (bookformat.html)

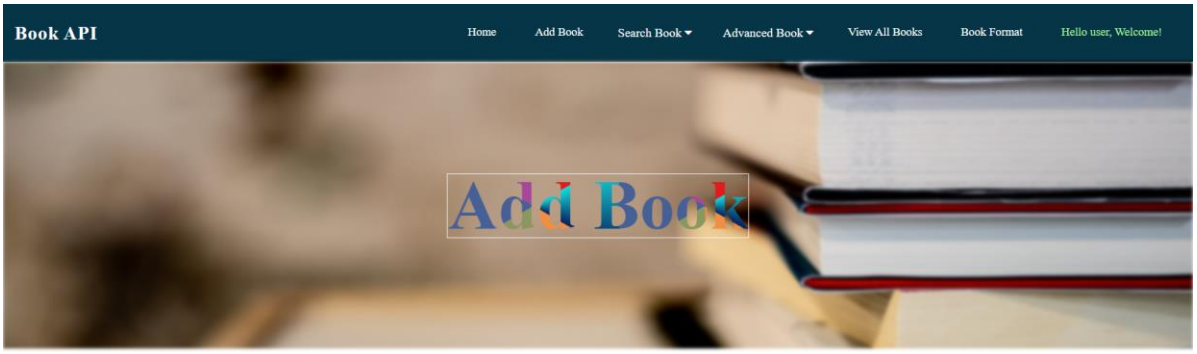
The function that is done when the search button is pressed is shown in figure 53. The *title* and *pub_year* variable are declared which is then concatenated with the url. An if statement is utilized to check whether the textboxes are empty. If they are empty, a message will be displayed in *red* that *Please fill all the textboxes*. Else another if statement is utilized to check if the book format is json or xml and if bookformat==json, the response will be displayed in json format else if bookformat==xml, the response will be displayed in xml format.

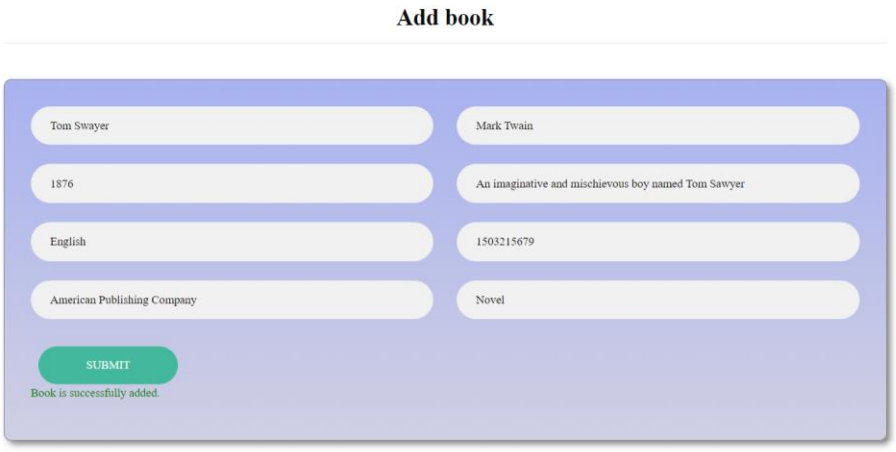
Testing

Add Book

Description: User should be able to add book.

Result:





Status: Pass

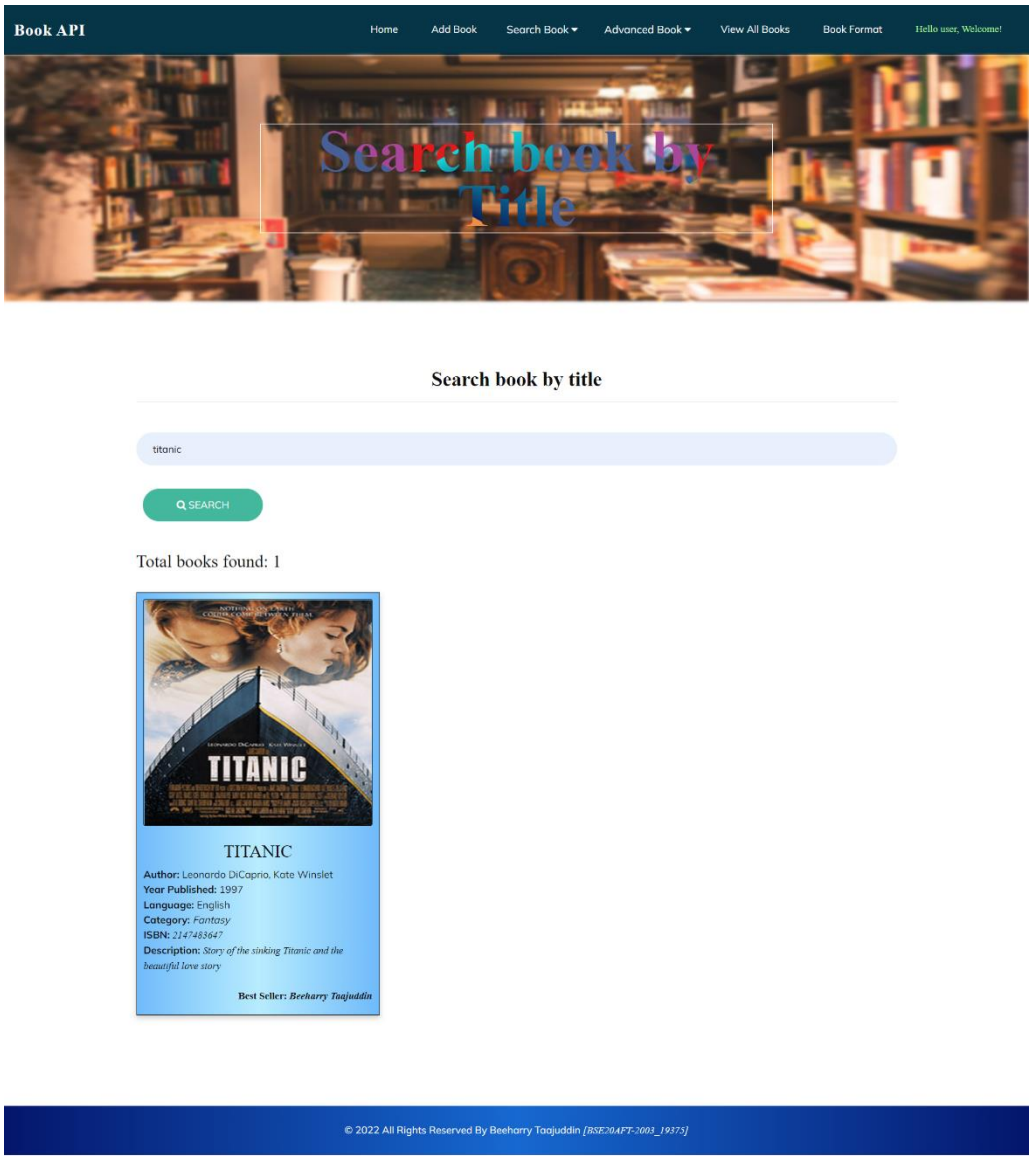
Post-Conditions: Book can be added successfully.

Search Book

By title

Description: Book title should be able to search by a user.

Result:



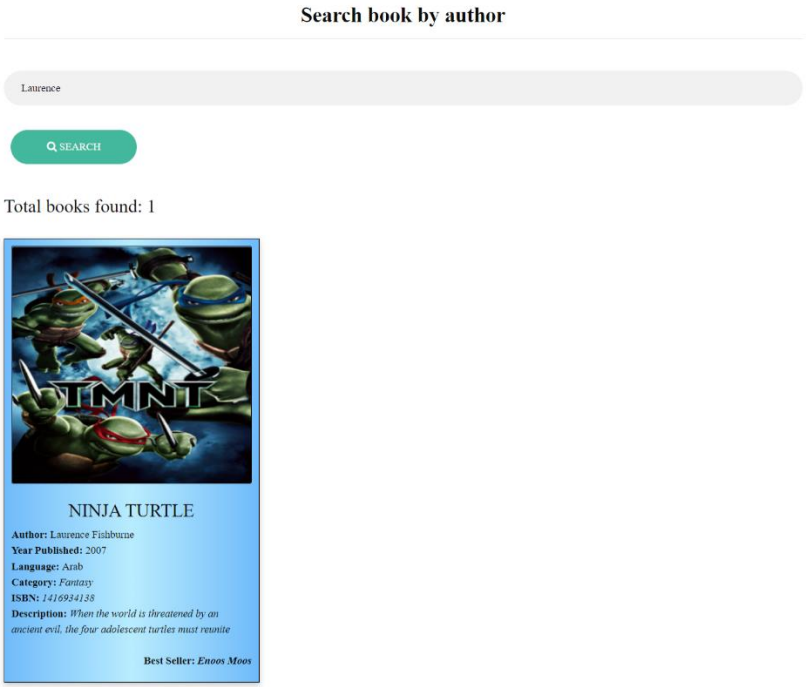
Status: Pass

Post-Conditions: Book title can be successfully searched.

By author

Description: Book author should be able to search by a user.

Result:



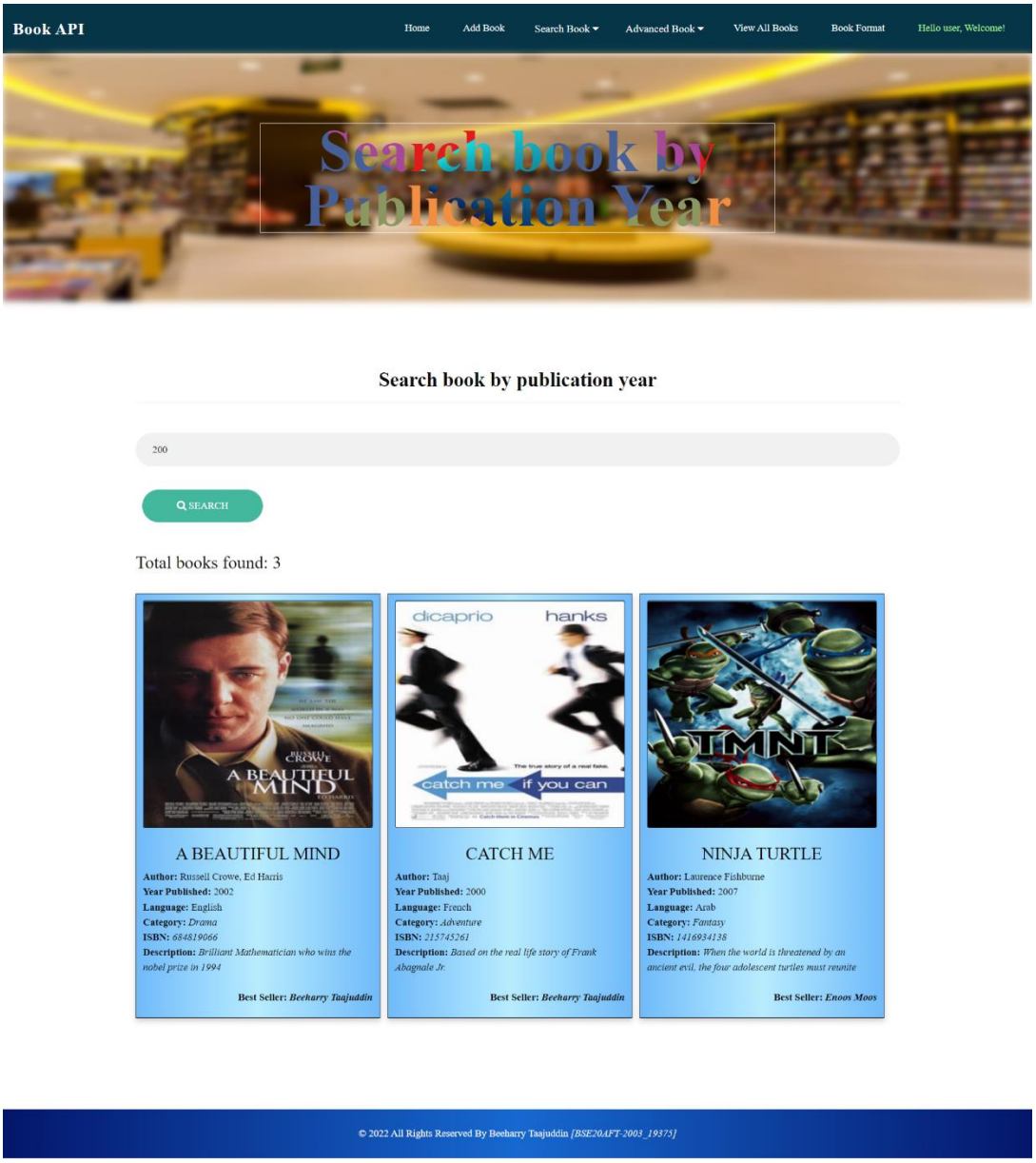
Status: Pass

Post-Conditions: Book author can be successfully searched.

By publication year

Description: Book publication year should be able to search by a user.

Result:



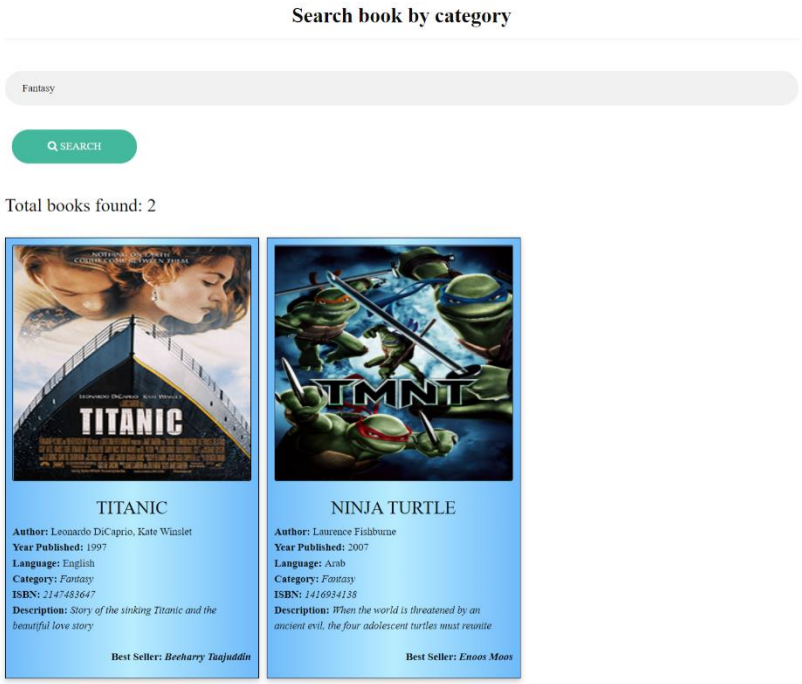
Status: Pass

Post-Conditions: Book publication year can be successfully searched.

By category

Description: Book category should be able to search by a user.

Result:



Status: Pass

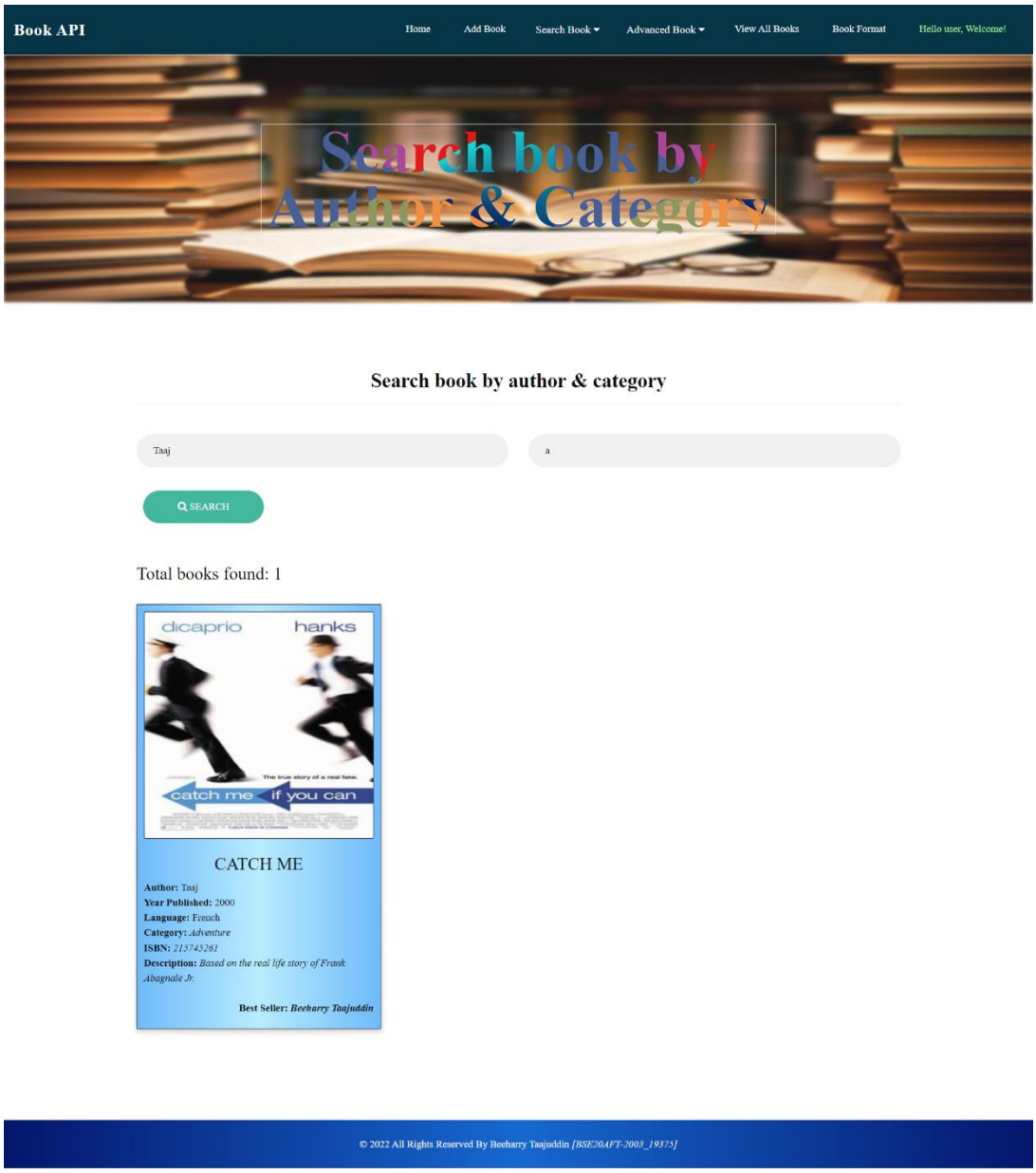
Post-Conditions: Book category can be successfully searched.

Advanced Search Book

By author & category

Description: Book author and category should be able to search by a user.

Result:



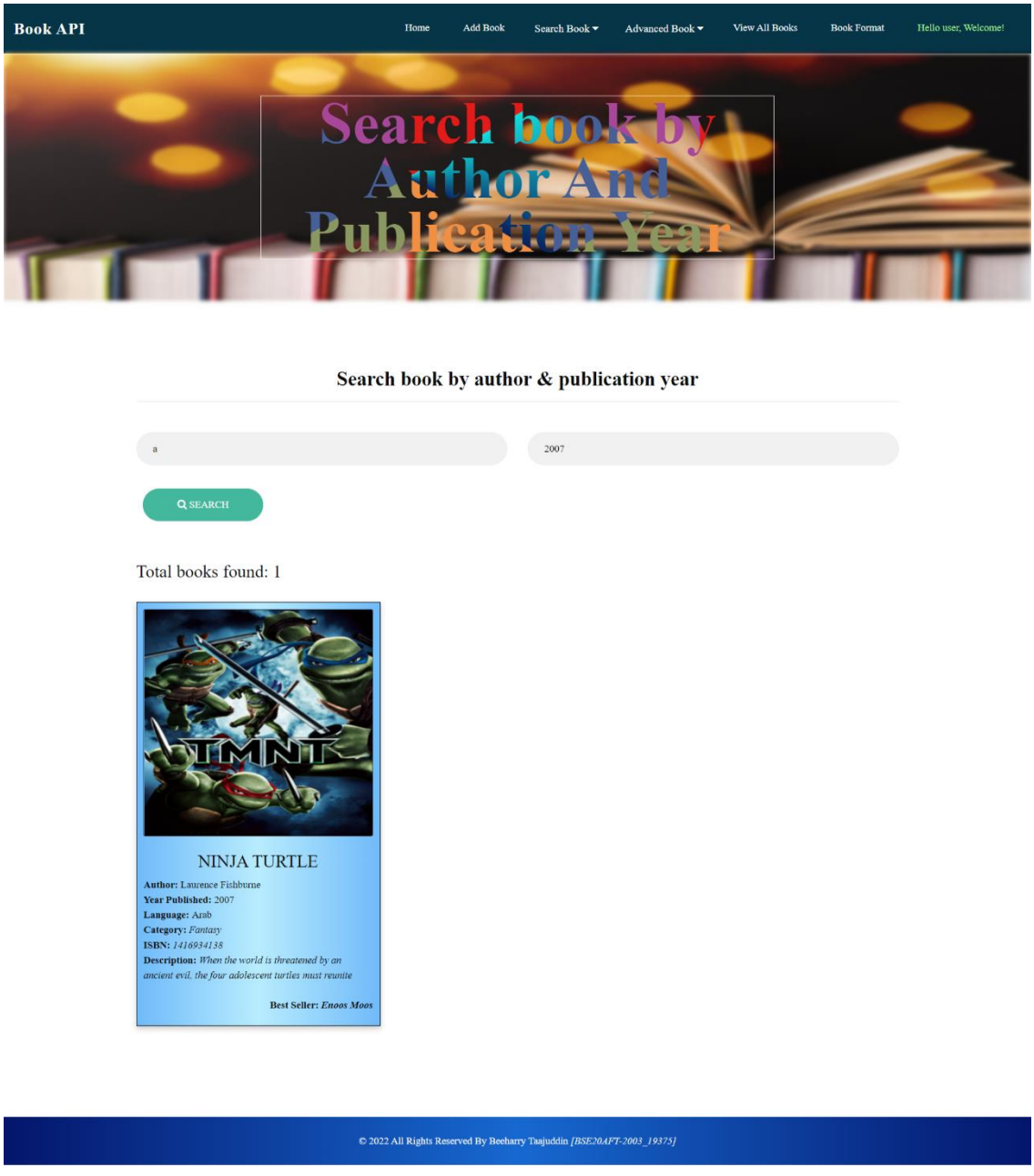
Status: Pass

Post-Conditions: Book author and category can be successfully searched.

By author & publication year

Description: Book author and publication year should be able to search by a user.

Result:



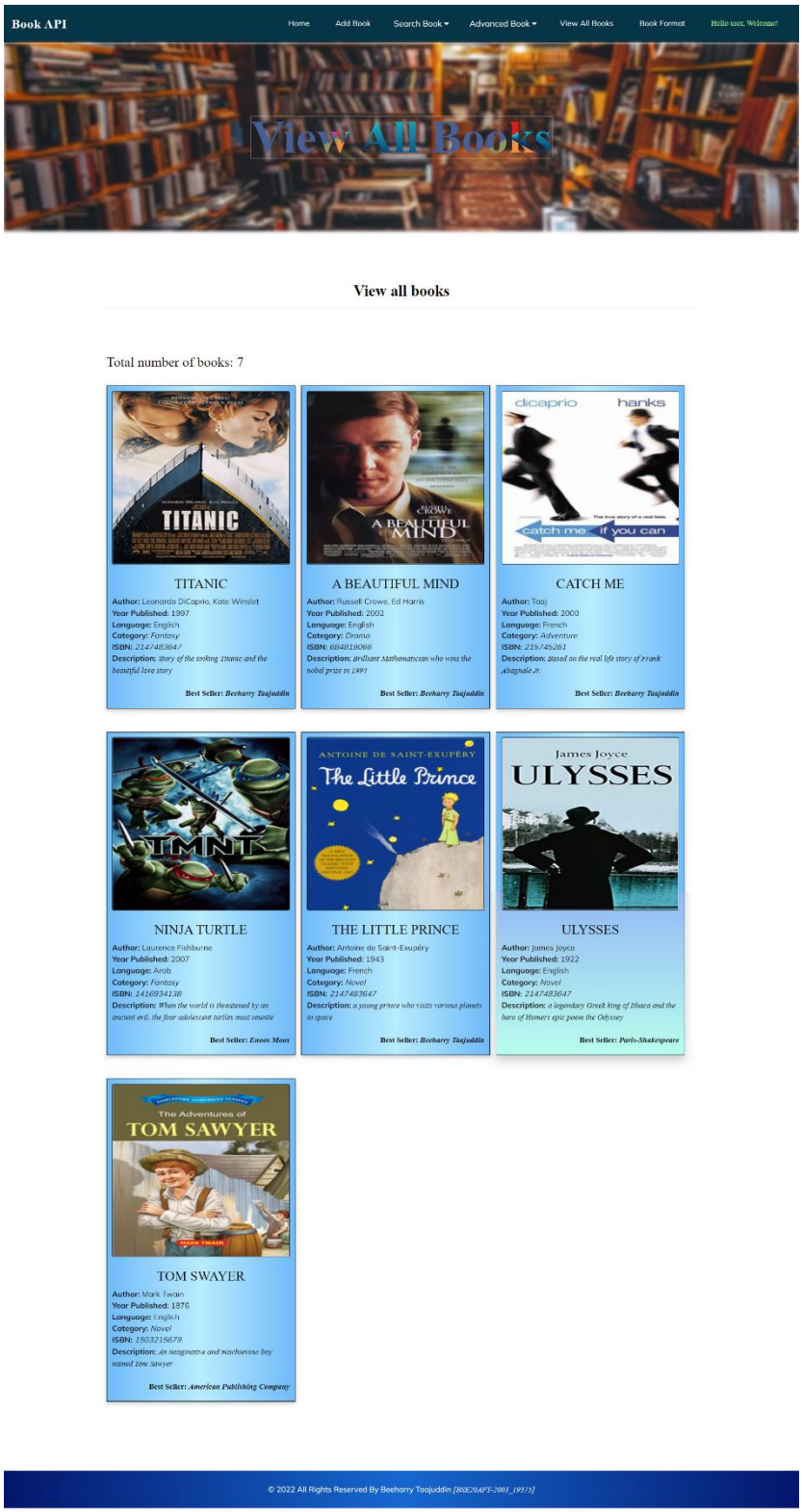
Status: Pass

Post-Conditions: Book author and publication year can be successfully searched.

View All Book

Description: User should be able to view all the books available in the bookstore.

Result:



Status: Pass

Post-Conditions: User can view all the books.

Book Format

Description: Users & programmers should be able to view book by json or xml format.

Result:

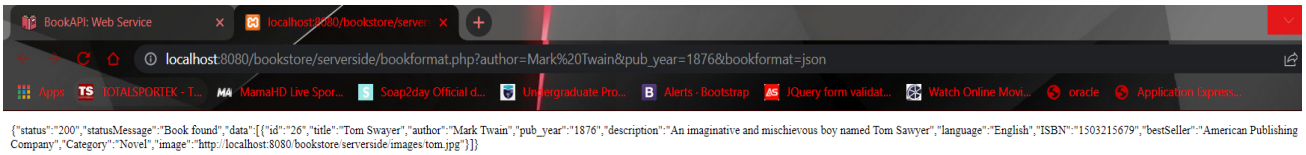


Figure 54 - Testing book format author & pub year in json

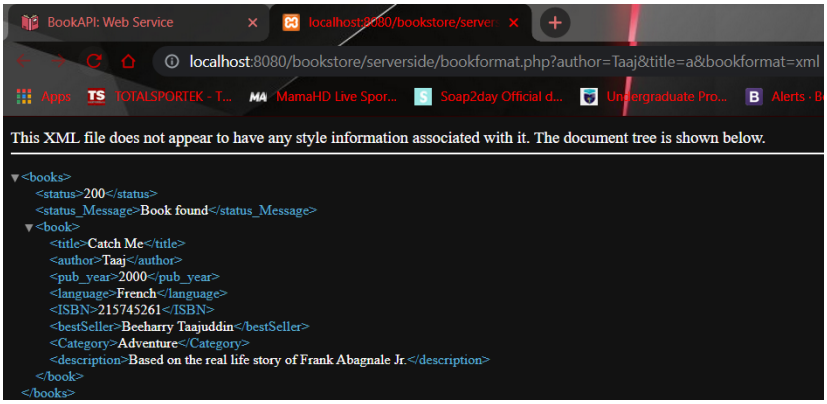


Figure 55 - Testing book format author & title in xml

Figure 56 - Testing book format title & pub year in json

Status: Pass

Post-Conditions: Users & programmers can view book in json or xml format.