



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)

Book API: REST Web Service (Bookstore)

For All Your Reading Needs

You can search your desire books by title, author, year, category.
Advanced search also available by author and category, author and year.
.. Beeharry Taajuddin..

Books Categories

These are the categories of books we have on the website.

 Drama fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using	 Science fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using	 History fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using
 Novel fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using	 Adventure fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using	 Fantasy fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using

About Our Book API
REST Web Service Bookstore

Here, at our book API REST Web Service Bookstore you can have variety of beautiful books. You can even search for your desire books and advanced search also available. You can also add your own book.

Hope you have enjoy these beautiful books we have.

© 2022 All Rights Reserved By Beeharry Taajuddin [BSE04FT-2003_19375]

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)

The screenshot shows the 'Add Book' page of a web application. The header is dark blue with white text. It includes a logo 'Book API', navigation links for 'Home', 'Add Book', 'Search Book', 'Advanced Book', 'View All Books', 'Book Format', and a personalized greeting 'Hello user, Welcome!'. The main title 'Add Book' is centered in a large, colorful font ('Add' in blue, 'Book' in red and orange) over a blurred background image of several books. Below the title is a light purple rectangular form containing six input fields arranged in two rows of three. The first row contains 'Enter book title' and 'Enter book author'. The second row contains 'Enter book publication year' and 'Enter book description'. The third row contains 'Enter book language' and 'Enter book isbn'. The fourth row contains 'Enter book best seller' and 'Enter book category'. At the bottom of the form is a green rounded rectangle button labeled 'SUBMIT'.

© 2022 All Rights Reserved By Beeharry Taajuddin (BSE20AFT-2003_19375)

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)

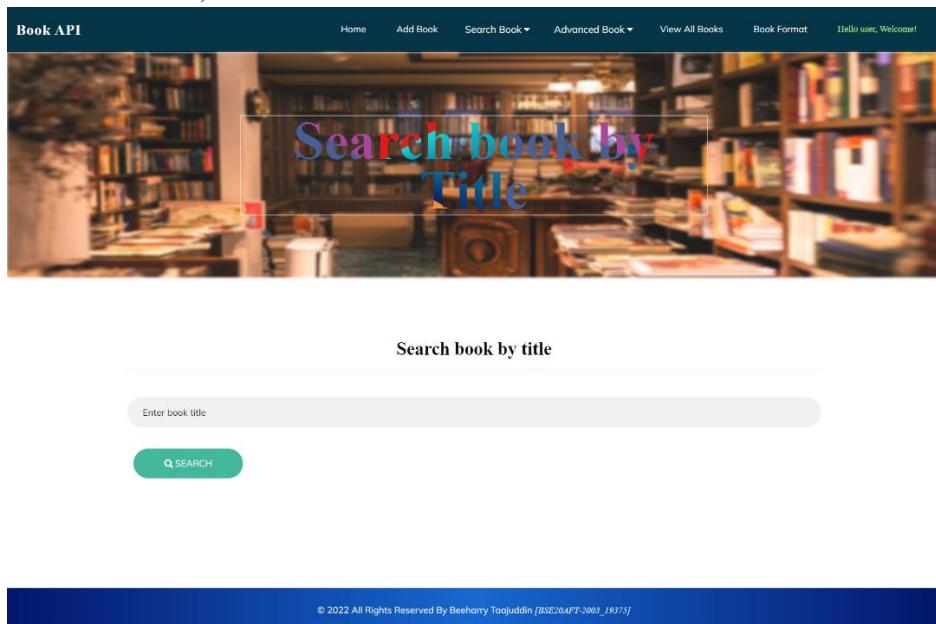


Figure 3 - Search by title (searchbooktitle.html(1))

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.

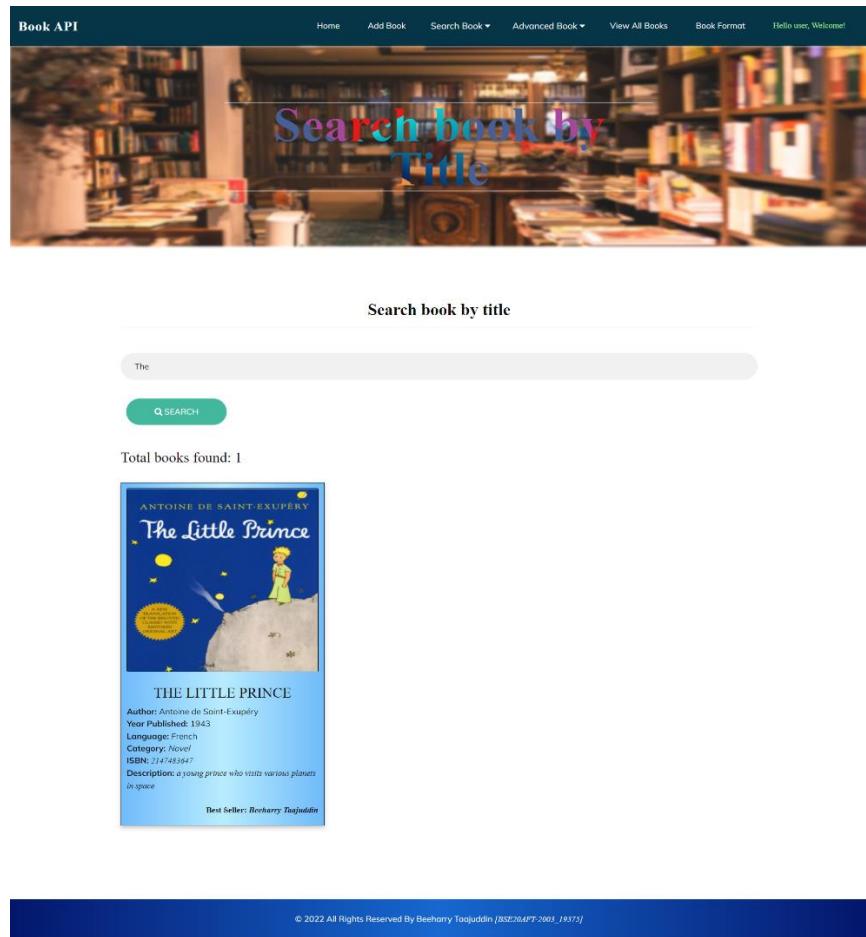


Figure 4 - Search by title (`searchbooktitle.html(2)`)

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`)

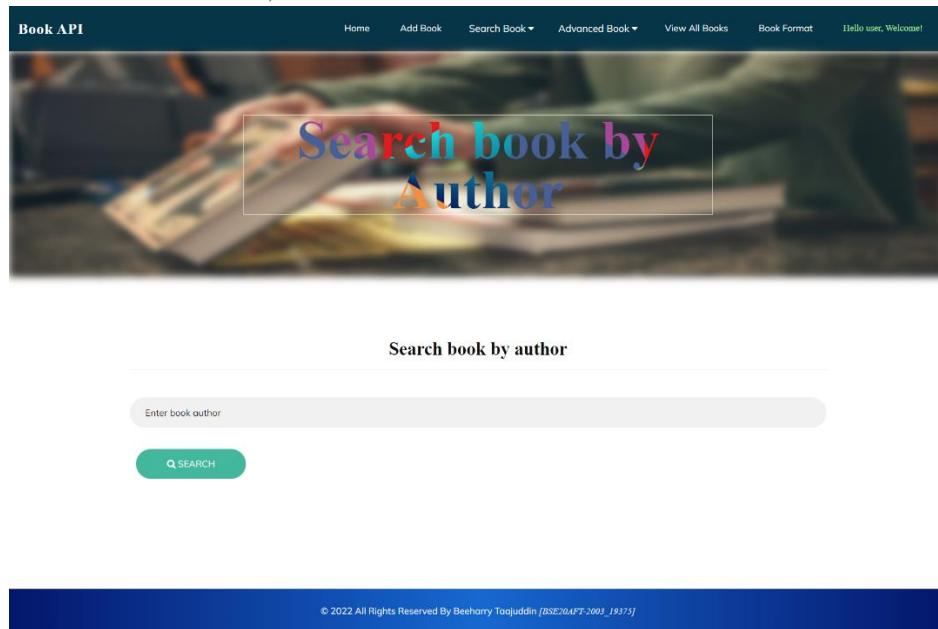


Figure 5 - Search by author (`searchbookauthor.html(1)`)

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)

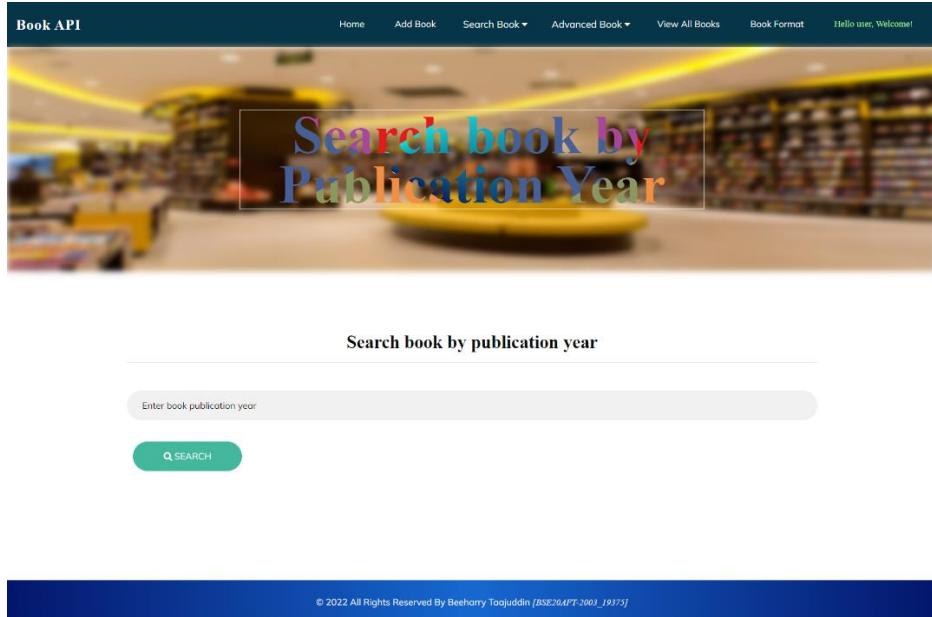


Figure 7 - Search by pub year (searchbookyear.html(1))

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

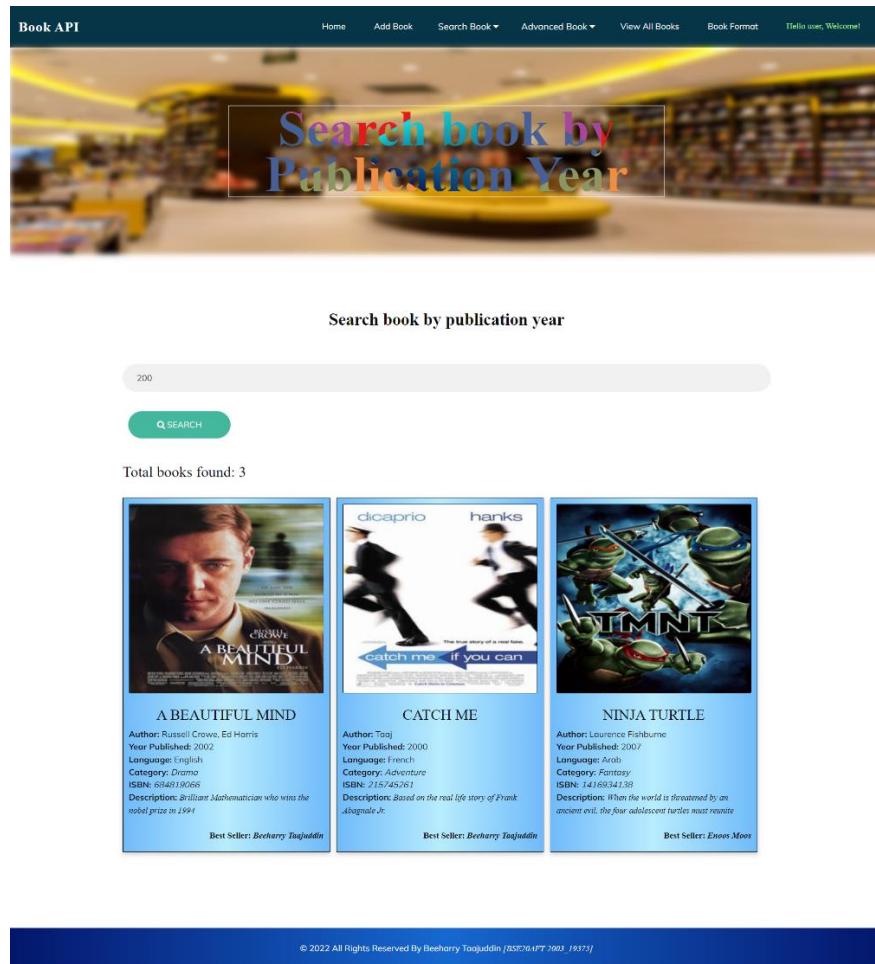


Figure 8 - Search by pub year (searchbookyear.html(2))

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)

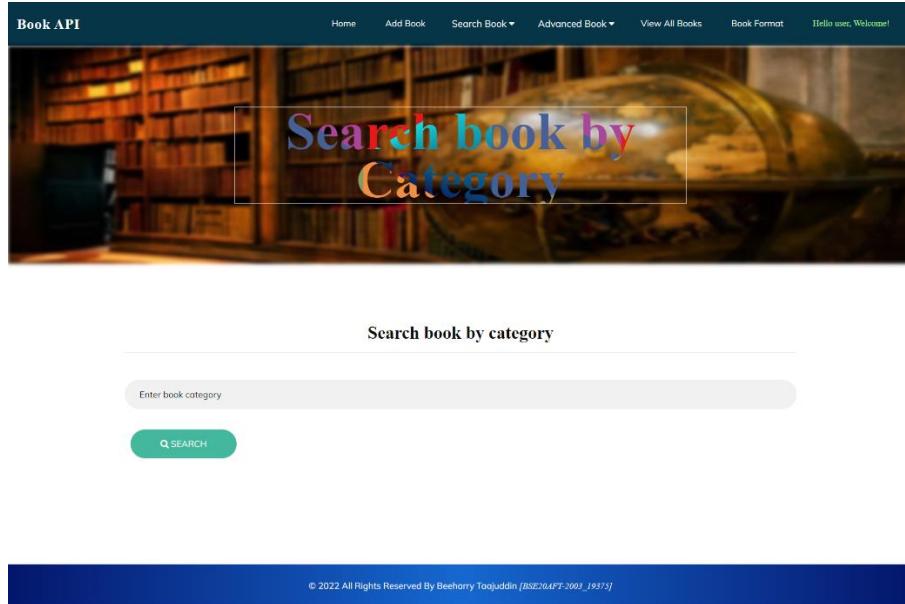


Figure 9 - Search by category (searchbookcat.html(1))

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

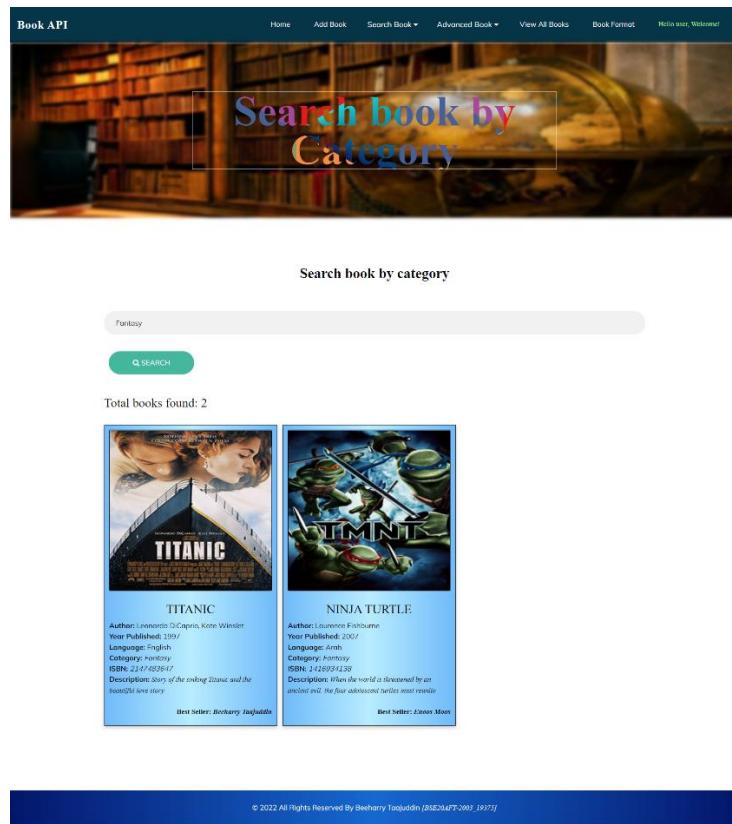


Figure 10 - Search by category (searchbookcat.html(2))

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)

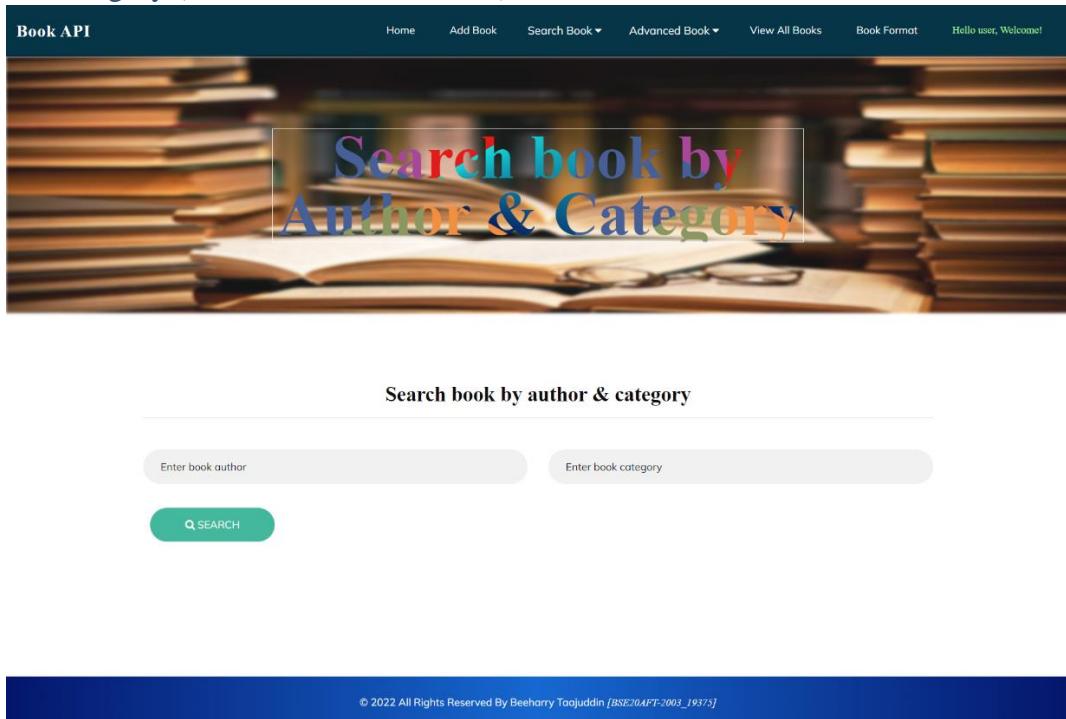


Figure 11 - Advanced search by author & category (advancedsearchAC.html(1))

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

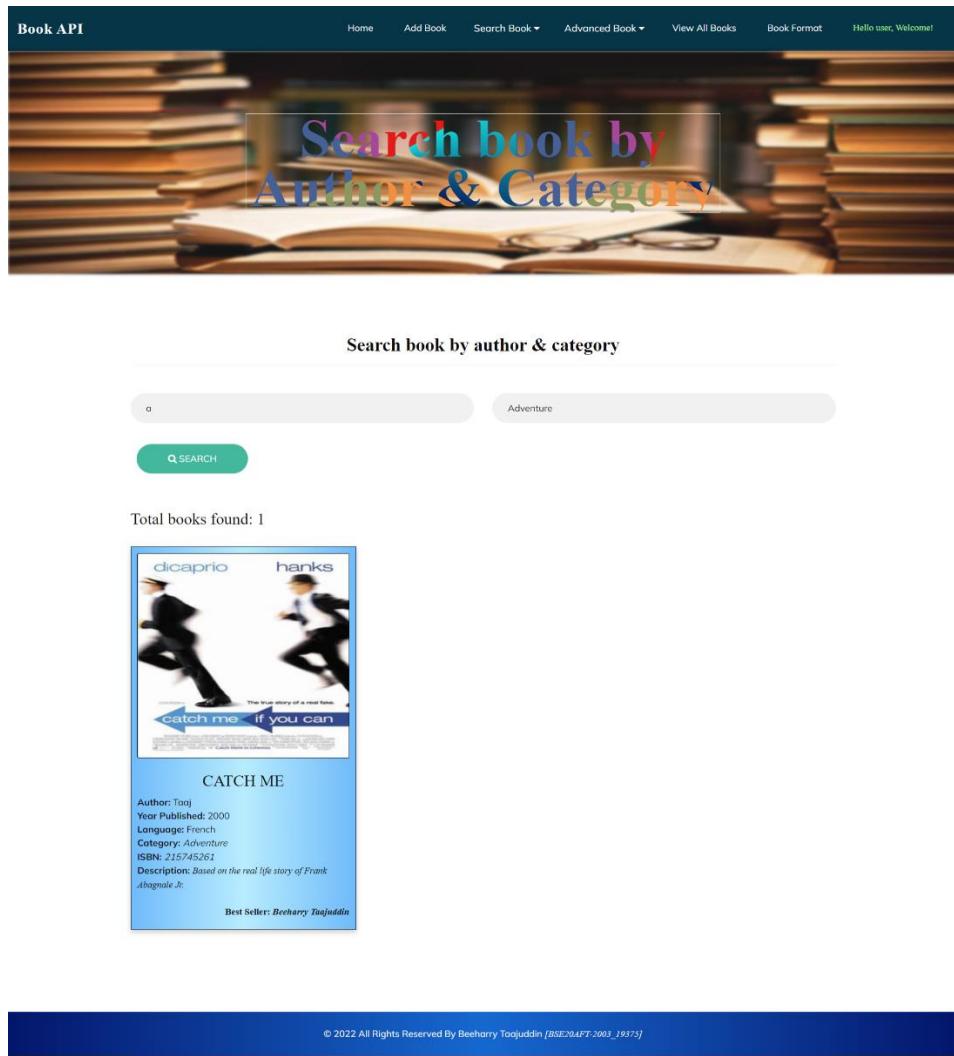


Figure 12 - Advanced search by author & category (advancedsearchAC.html(2))

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)

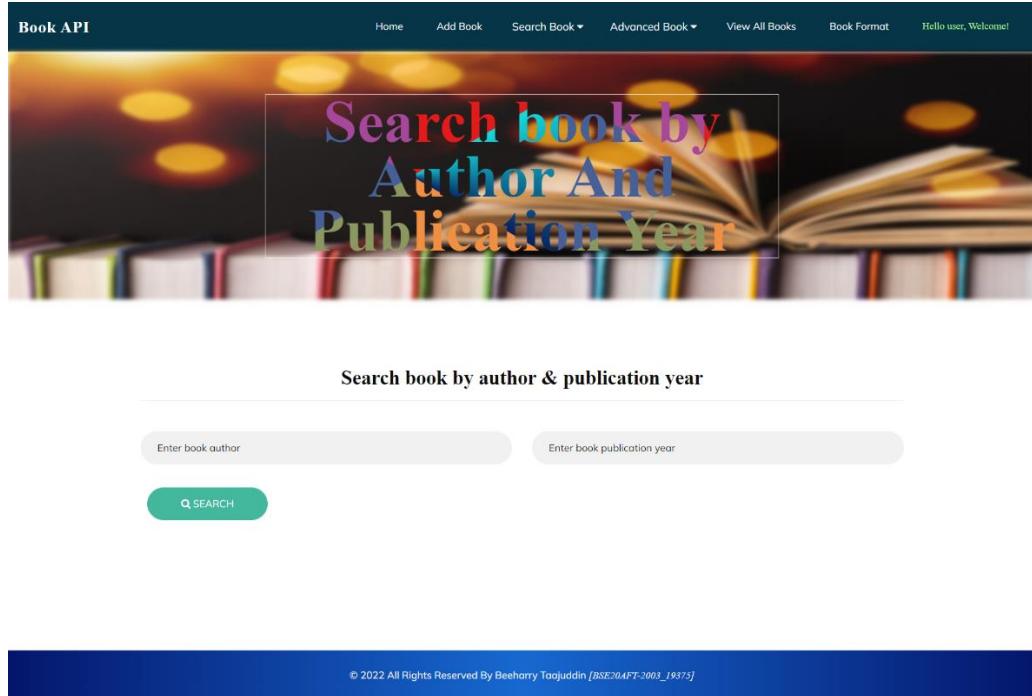


Figure 13 - Advanced search by author & pub year (advancedsearchAY.html(1))

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

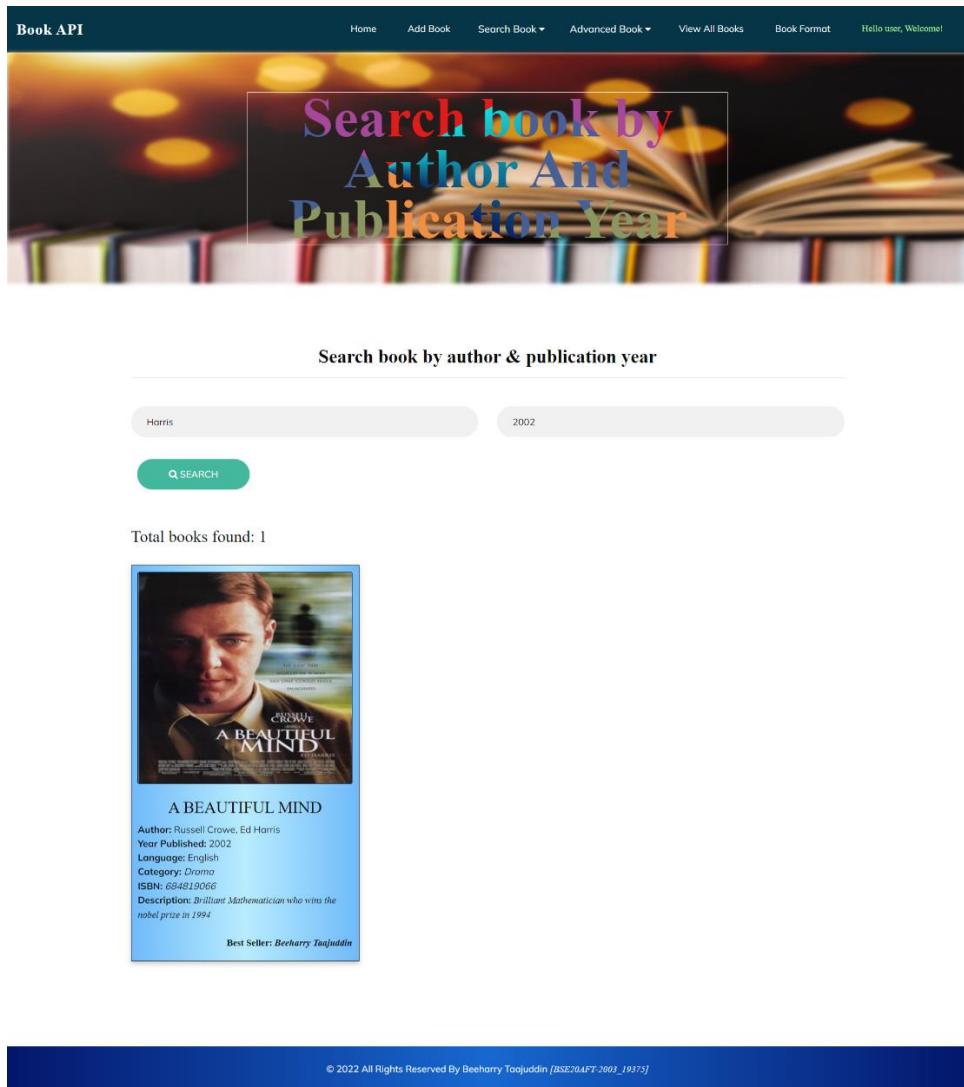


Figure 14 - Advanced search by author & pub year (advancedsearchAY.html(2))

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)

Book API

Home Add Book Search Book Advanced Book View All Books Book Format Hello user, Welcome!

Total number of books: 6

TITANIC Author: Leonardo DiCaprio, Kate Winslet Year Published: 1997 Language: English Category: Fantasy ISBN: 2147483647 Description: Story of the sinking Titanic and the beautiful love story	A BEAUTIFUL MIND Author: Russell Crowe, Ed Harris Year Published: 2002 Language: English Category: Drama ISBN: 684819066 Description: Brilliant Mathematician who wins the nobel prize in 1994	CATCH ME Author: Tom Hanks Year Published: 2000 Language: French Category: Adventure ISBN: 215745261 Description: Based on the real life story of Frank Abagnale Jr.
NINJA TURTLE Author: Laurence Fishburne Year Published: 2007 Language: Arab Category: Fantasy ISBN: 1416934138 Description: When the world is threatened by an ancient evil, the four adolescent turtles must reunite	THE LITTLE PRINCE Author: Antoine de Saint-Exupéry Year Published: 1943 Language: French Category: Novel ISBN: 2147483647 Description: a young prince who visits various planets in space	ULYSSES Author: James Joyce Year Published: 1922 Language: English Category: Novel ISBN: 2147483647 Description: a legendary Greek king of Ithaca and the hero of Homers epic poem the Odyssey

© 2022 All Rights Reserved By Beeharry Taajuddin [BSE20AFT-2003_19375]

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)

The figure consists of three vertically stacked screenshots of a web application interface. At the top is a banner with the text "Book Format by Xml & Json" overlaid on an image of an open book. Below the banner are three separate search forms:

- Book format by Author & Publication Year:** Contains fields for "Author" (set to "Taaj"), "Publication Year" (set to "2"), and a dropdown for "JSON Format".
- Book format by Author & Title:** Contains fields for "Enter book author" (set to "Taaj") and "Enter book title".
- Book format by Title & Publication Year:** Contains fields for "Enter book title" and "Enter book publication year".

A dark blue footer bar at the bottom contains the text "© 2022 All Rights Reserved By Beeharry Taajuddin JSE200APT-2003_193757".

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.

A screenshot of a browser window showing the JSON output of a book search. The URL in the address bar is "localhost:8080/bookstore/serverside/bookformat.php?author=Taaj&pub_year=2&bookformat=json". The page content is a JSON object:

```
{"status": "200", "statusMessage": "Book found", "data": [{"id": "3", "title": "Catch Me", "author": "Taaj", "pub_year": "2000", "description": "Based on the real life story of Frank Abagnale Jr.", "language": "French", "ISBN": "215745261", "bestSeller": "Beeharry Taajuddin", "Category": "Adventure", "image": "http://localhost:8080/bookstore/serverside/images/catchme.jpg"}]}
```

Figure 17 - Json book format

Figure 17 shows the json book format.

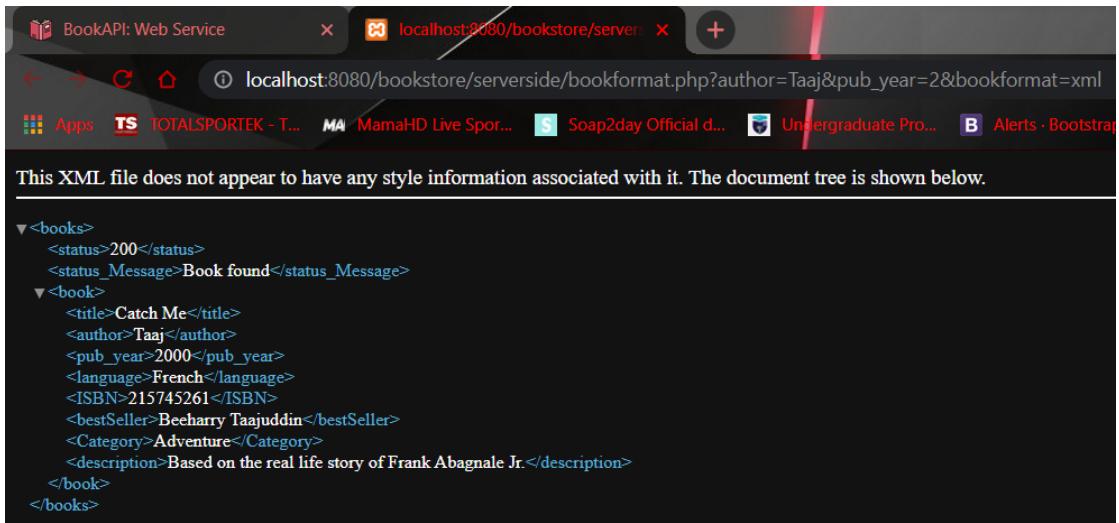


Figure 18 - Xml book format

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.

```

120 function getBookByAuthorCat($author,$category){
121     //connection object $conn holds the connection to the Database
122     $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
123
124     //TODO
125     $query = "select * from books where author LIKE '%$author%' AND Category LIKE '%$category%'";
126     $result = mysqli_query($conn, $query);
127     $book_details = null;
128
129     if($result){
130         $data=array();
131         while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
132             $data[]=$row;
133         }
134     return $data;
135 }
136
137 }
138
139 function getBookByAuthorYear($author,$year){
140     //connection object $conn holds the connection to the Database
141     $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
142
143     //TODO
144     $query = "select * from books where author LIKE '%$author%' AND pub_year LIKE '%$year%'";
145     $result = mysqli_query($conn, $query);
146     $book_details = null;
147
148     if($result){
149         $data=array();
150         while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
151             $data[]=$row;
152         }
153     return $data;
154 }
155
156 }

```

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.

```

159 function getAllBookByData(){
160     //connection object $conn holds the connection to the Database
161     $conn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: ' . mysqli_connect_error());
162
163     //TODO
164     $query = "select * from books";
165     $result = mysqli_query($conn, $query);
166     $book_details = null;
167
168     if($result){
169         $data=array();
170         while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
171             $data[]=$row;
172         }
173     return $data;
174 }

```

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.

```

178     function addBook($title, $author, $pub_year, $description, $language, $ISBN, $bestSeller, $Category){
179         //connection object $conn holds the connection to the Database
180         $dbconn = mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL: '.mysqli_connect_error());
181         $image="http://localhost:8080/bookstore/serverside/images/bookcover.jpg";
182
183         //TODO
184         $query = "INSERT INTO books (title, author, pub_year, description, language, ISBN, bestSeller, Category, image)VALUES ('".$title."', '".$author."', '".$$.
185         pub_year."', '".$description."', '".$language."', '".$ISBN."', '".$bestSeller."', '".$Category."', '".$image."')";
186         if (mysqli_query($dbconn, $query)) {
187             echo "Book is successfully added.";
188         } else {
189             echo "Error: " . $query . "" . mysqli_error($dbconn);
190         }
191         $dbconn->close();
192     }

```

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

```
1 <?php
2
3 include("bookFunctions.php");
4
5 $title = null;
6 $author = null;
7 $pub_year = null;
8 $description = null;
9 $language = null;
10 $ISBN = null;
11 $bestSeller = null;
12 $Category = null;
13
14 if(!empty($_GET['title']) && !empty($_GET['author']) && !empty($_GET['pub_year']) && !empty($_GET['description']) && !empty($_GET['language']) &&
15 !empty($_GET['ISBN']) && !empty($_GET['bestSeller']) && !empty($_GET['Category']))
16 {
17
18     $title = $_GET['title'];
19     $author = $_GET['author'];
20     $pub_year = $_GET['pub_year'];
21     $description = $_GET['description'];
22     $language = $_GET['language'];
23     $ISBN = $_GET['ISBN'];
24     $bestSeller = $_GET['bestSeller'];
25     $Category = $_GET['Category'];
26
27     if($title != null && $author != null && $pub_year != null && $description != null && $language != null && $ISBN != null && $bestSeller != null && $Category
28         != null)
29     {
30         addBook($title, $author, $pub_year, $description, $language, $ISBN, $bestSeller, $Category);
31     }
32 }
33 else{
34     echo "All text fields are required!!";
35 }
36
37
38 ?>
```

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()`.

```
340     <div class="backg">
341         <div class="row">
342             <div class="col-md-6"><input type="text" id="btitle" placeholder="Enter book title" /></div>
343             <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
344             <div class="col-md-6"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
345             <div class="col-md-6"><input type="text" id="bdes" placeholder="Enter book description" /></div>
346             <div class="col-md-6"><input type="text" id="blang" placeholder="Enter book language" /></div>
347             <div class="col-md-6"><input type="text" id="bisbn" placeholder="Enter book isbn" /></div>
348             <div class="col-md-6"><input type="text" id="bbseller" placeholder="Enter book best seller" /></div>
349             <div class="col-md-6"><input type="text" id="bcat" placeholder="Enter book category" /></div>
350         </div>
351         <div class="btn-box">
352             <button onclick="Submit()" >SUBMIT</button>
353         </div>
354         <p id="msg"></p>
355     </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.

```

219 <script>
220     function Submit() {
221         var title = document.getElementById('btitle').value;
222         var author = document.getElementById('bauthor').value;
223         var pub_year = document.getElementById('byear').value;
224         var description = document.getElementById('bdes').value;
225         var language = document.getElementById('blang').value;
226         var ISBN = document.getElementById('bisbn').value;
227         var bestSeller = document.getElementById('bseller').value;
228         var Category = document.getElementById('bcat').value;
229
230         if(title == "" || author == "" || pub_year == "" || description == "" || language == "" || ISBN == "" || bestSeller == "" || Category == ""){
231             msg.textContent = "Please fill all the textboxes!";
232             msg.style.color = "darkred";
233             return false;
234         }
235         else {
236             var url = "http://localhost:8080/bookstore/serverside/addbook.php?title="+title+"&author="+author+"&pub_year="+pub_year+"&description="+description+
237             "&language="+language+"&ISBN="+ISBN+"&bestSeller="+bestSeller+"&Category="+Category;
238
239             var xhttp = new XMLHttpRequest();
240
241             xhttp.open("POST", url, true);
242             xhttp.send(title, author, pub_year, description, language, ISBN, bestSeller, Category);
243             //window.location.href = "addbook.html";
244
245             msg.textContent = "Book is successfully added.";
246             msg.style.color = "green";
247
248             var title = "";
249             var author = "";
250             var pub_year = "";
251             var description = "";
252             var language = "";
253             var ISBN = "";
254             var bestSeller = "";
255             var Category = "";
256         }
257     }</script>

```

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

The *Submit()* function is shown in 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 is 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

```

323         <b>Search book by title</b>
324         </h3><hr style="margin-bottom: 45px;" />
325         <div>
326             <input type="text" id="btitle" placeholder="Enter book title" />
327         </div>
328         <div class="btn-box">
329             <button onclick="SearchByTitle()" >
330                 <i class="fa fa-search" aria-hidden="true"></i> SEARCH
331             </button>
332         </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 is called when the button is clicked. The **id="displayBtitle"** is used to display the book results.

```

207 <script>
208     function SearchByTitle(){
209         var title = document.getElementById("btitle").value;
210         var url="http://localhost:8080/bookstore/serverside/index.php?title="+title;
211
212         var xmlhttp = new XMLHttpRequest();
213         xmlhttp.onreadystatechange = function () {
214             if (this.readyState == 4 && this.status == 200) {
215                 var book = JSON.parse(this.responseText);
216
217                 var result="";
218                 result+="  
<h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";
219
220                 for(var i = 0; i < book.data.length; i++){
221                     result+="

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



```

322 <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
323 | Search book by author
324 |</h3><hr style="margin-bottom: 45px;" />
325 <div>
326 <input type="text" id="bauthor" placeholder="Enter book author" />
327 </div>
328 <div class="btn-box">
329 <button onclick="SearchByAuthor()" >
330 <i class="fa fa-search" aria-hidden="true"></i> SEARCH
331 </button>
332 </div>
<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.



27 | Page


```

```

207 <script>
208     function SearchByAuthor(){
209         var author = document.getElementById("bauthor").value;
210         var url="http://localhost:8080/bookstore/serverside/index.php?author="+author;
211
212         var xmlhttp = new XMLHttpRequest();
213         xmlhttp.onreadystatechange = function () {
214             if (this.readyState == 4 && this.status == 200) {
215                 var book = JSON.parse(this.responseText);
216
217                 var result="";
218                 result+="  
<h3 style='font-family: sans-serif;'>"+Total books found: "+book.data.length+"</h3><br/>";
219
220                 for(var i = 0; i < book.data.length; i++){
221                     result+="

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



```

322 <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
323 Search book by publication year
324 </h3><hr style="margin-bottom: 45px;" />
325 <div>
326 <input type="text" id="byear" placeholder="Enter book publication year" />
327 </div>
328 <div class="btn-box">
329 <button onclick="SearchByYear()" >
330 <i class="fa fa-search" aria-hidden="true"></i> SEARCH
331 </button>
332 </div>
<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.



28 | Page


```

```

207 <script>
208     function SearchByYear(){
209         var pub_year = document.getElementById("byear").value;
210         var url="http://localhost:8080/bookstore/serverside/index.php?pub_year="+pub_year;
211
212         var xmlhttp = new XMLHttpRequest();
213         xmlhttp.onreadystatechange = function () {
214             if (this.readyState == 4 && this.status == 200) {
215                 var book = JSON.parse(this.responseText);
216
217                 var result="";
218                 result+="  
 <h3 style='font-family: sans-serif;'>"+Total books found: "+book.data.length+"</h3><br/>";
219
220                 for(var i = 0; i < book.data.length; i++){
221                     result+="  
<h3 style='text-transform: uppercase; text-align: center; font-family: sans-serif;'>"+book.data[i].title+"</h3>"+"  
<b>Author: </b>"+book.data[i].author
222                     +"<br/><b>Year Published: </b>"+book.data[i].pub_year
223                     +"<br/><b>Language: </b>"+book.data[i].language
224                     +"<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>+book.data[i].description+</i>"+"  
<br/><b>Best Seller: </b> "+<i>+book.data[i].bestSeller+</i>"+<br/>"+"
```

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

```

322             <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
323                 <b>Search book by category</b>
324             </h3><hr style="margin-bottom: 45px;" />
325             <div>
326                 <input type="text" id="bcat" placeholder="Enter book category" />
327             </div>
328             <div class="btn-box">
329                 <button onclick="SearchByCat()" >
330                     <i class="fa fa-search" aria-hidden="true"></i> SEARCH
331                 </button>
332             </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.

```

207 <script>
208     function SearchByCat(){
209         var Category = document.getElementById("bcat").value;
210         var url="http://localhost:8080/bookstore/serverside/index.php?Category="+Category;
211
212         var xmlhttp = new XMLHttpRequest();
213         xmlhttp.onreadystatechange = function () {
214             if (this.readyState == 4 && this.status == 200) {
215                 var book = JSON.parse(this.responseText);
216
217                 var result="";
218                 result+="  
<h3 style='font-family: sans-serif;'>"+Total books found: "+book.data.length+"</h3><br/>";
219
220                 for(var i = 0; i < book.data.length; i++){
221                     result+="<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
222                     +"<br><b>Year Published: </b>"+book.data[i].pub_year
223                     +"<br><b>Language: </b>"+book.data[i].language
224                     +"<br><b>Category: </b>"+<i>+book.data[i].Category+</i>"+"  
<b>ISBN: </b>"+<i>+book.data[i].ISBN+</i>"+"  
<b>Description: </b>"+<i>+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>";
225                 }
226                 document.getElementById("displayBcat").innerHTML = result;
227
228             }else{
229                 document.getElementById("displayBcat").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
230             }
231         };
232         xmlhttp.open("GET", url, true);
233         xmlhttp.send();
234     }
235     </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

```

331     <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
332         <b>Search book by author & category</b>
333     </h3><hr style="margin-bottom: 45px;" />
334     <div class="row">
335         <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
336         <div class="col-md-6"><input type="text" id="bcat" placeholder="Enter book category" /></div>
337     </div>
338     <div class="btn-box">
339         <button onclick="SearchByAuthorCat()" >
340             <i class="fa fa-search" aria-hidden="true"></i> SEARCH
341         </button>
342     </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.

```

207 <script>
208     function SearchByAuthorCat(){
209         var author = document.getElementById("bauthor").value;
210         var Category = document.getElementById("bcat").value;
211         if(author == "" || Category == ""){
212             msg.textContent = "Author and Category are required!";
213             msg.style.color = "darkred";
214             return false;
215         }
216     }else{
217
218         var url="http://localhost:8080/bookstore/serverside/index.php?author="+author+"&category="+Category;
219
220         var xmlhttp = new XMLHttpRequest();
221         xmlhttp.onreadystatechange = function () {
222             if (this.readyState == 4 && this.status == 200) {
223                 var book = JSON.parse(this.responseText);
224
225                 var result="";
226                 result+="  
<h3 style='font-family: sans-serif;'>"+"Total books found: "+book.data.length+"</h3><br/>";
227
228                 for(var i = 0; i < book.data.length; i++){
229                     result+="

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



```

331 <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
332 Search book by author & publication year
333 </h3><hr style="margin-bottom: 45px;" />
334 <div class="row">
335 <div class="col-md-6"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
336 <div class="col-md-6"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
337 </div>
338 <div class="btn-box">
339 <button onclick="SearchByAuthorYear()" >
340 <i class="fa fa-search" aria-hidden="true"></i> SEARCH
341 </button>
342 </div>
<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.



31 | Page


```

```

207 <script>
208     function SearchByAuthorYear(){
209         var author = document.getElementById("bauthor").value;
210         var pub_year = document.getElementById("byear").value;
211         if(author == "" || pub_year == ""){
212             msg.textContent = "Author and Publication Year are required!";
213             msg.style.color = "darkred";
214             return false;
215         }
216     }else{
217         var url="http://localhost:8080/bookstore/serverside/index.php?author="+author+"&pub_year="+pub_year;
218
219         var xmlhttp = new XMLHttpRequest();
220         xmlhttp.onreadystatechange = function () {
221             if (this.readyState == 4 && this.status == 200) {
222                 var book = JSON.parse(this.responseText);
223
224                 var result="";
225                 result+="  


### "+"Total books found: "+book.data.length+"

  
"
226
227                 for(var i = 0; i < book.data.length; i++){
228                     result+="

<"+"img class='card-img' src='"+book.data[i].image+ "' style='width:100%;height:340px;'' alt='No Image'"+">"+"  
  


### 

Author: "+book.data[i].author
229                     +"<br><b>Year Published:</b> "+book.data[i].pub_year
230                     +"<br><b>Language:</b> "+book.data[i].language
231                     +"<br><b>Category:</b> "+<i>+book.data[i].Category+</i>
232                     +"<br><b>ISBN:</b> "+<i>+book.data[i].ISBN+</i>
233                     +"<br><b>Description:</b> "+<i style='font-family: sans-serif;'>+book.data[i].description+</i>
234                     +"<br><br><b style='font-family: sans-serif; float: right;'>Best Seller: "<i>+<i>+book.data[i].bestSeller+</i>"+</b>
235                     +"</div>";
236                 }
237                 document.getElementById("displayBAuYear").innerHTML = result;
238             }else{
239                 document.getElementById("displayBAuYear").innerHTML = "<span style='color:red; font-weight:bold;'>No book found!</span>";
240             }
241         };
242         xmlhttp.open("GET", url, true);
243         xmlhttp.send();
244     }
245 },


```

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

```

315         <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
316             <b>View all books</b>
317         </h3><hr style="margin-bottom: 45px;" />
318         <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"**.

```

207 <script>
208     var url="http://localhost:8080/bookstore/serverside/index.php?data[]";
209
210     var xmlhttp = new XMLHttpRequest();
211     xmlhttp.onreadystatechange = function () {
212         var book = JSON.parse(this.responseText);
213
214         var result="";
215         result+="  
 <h3 style='font-family: sans-serif;'>"+"Total number of books: "+book.data.length+"</h3><br/>";
216
217         for(var i = 0; i < book.data.length; i++){
218             result+="

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.



33 | Page


```

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

```

376      <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
377          <b>Book format by Author & Publication Year</b>
378      </h3><hr style="margin-bottom: 45px;" />
379      <div class="row">
380          <div class="col-md-4"><input type="text" id="bauthor" placeholder="Enter book author" /></div>
381          <div class="col-md-4"><input type="text" id="byear" placeholder="Enter book publication year" /></div>
382          <div class="col-md-4">
383              <select name="format" class="txtdrop" id="AYformat">
384                  <option class="txtdrop1" value="">Select format</option>
385                  <option class="txtdrop1" value="json">JSON Format</option>
386                  <option class="txtdrop1" value="xml">XML Format</option>
387              </select>
388          </div>
389      </div>
390      <div class="btn-box">
391          <button onclick="SearchByAuthorYearFormat1()" >
392              <i class="fa fa-search" aria-hidden="true"></i> SEARCH
393          </button>
394      </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.

```

234 <script>
235     function SearchByAuthorYearFormat1(){
236         var author = document.getElementById("bauthor").value;
237         var pub_year = document.getElementById("byear").value;
238         var bookformat = document.getElementById("AYformat").value;
239         if(author == "" || pub_year == "" || bookformat == ""){
240             msg.textContent = "Please fill all the textboxes!"
241             msg.style.color = "darkred"
242             return false;
243         }else{
244             if (bookformat=='json'){
245                 window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&pub_year="+pub_year+"&bookformat="+bookformat);
246             }else if(bookformat=='xml'){
247                 window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&pub_year="+pub_year+"&bookformat="+bookformat);
248             }
249         }
250     }
251 }
252 
```

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

```

403     <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
404         <b>Book format by Author & Title</b>
405     </h3><hr style="margin-bottom: 45px;" />
406     <div class="row">
407         <div class="col-md-4"><input type="text" id="bauthor1" placeholder="Enter book author" /></div>
408         <div class="col-md-4"><input type="text" id="btitle" placeholder="Enter book title" /></div>
409         <div class="col-md-4">
410             <select name="format" class="txtdrop" id="ATformat">
411                 <option class="txtdrop1" value="">Select format</option>
412                 <option class="txtdrop1" value="json">JSON Format</option>
413                 <option class="txtdrop1" value="xml">XML Format</option>
414             </select>
415         </div>
416     </div>
417     <div class="btn-box">
418         <button onclick="SearchByAuthorTitleFormat()" >
419             <i class="fa fa-search" aria-hidden="true"></i> SEARCH
420         </button>
421     </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 *ATformat* 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.

```

255     function SearchByAuthorTitleFormat(){
256         var author = document.getElementById("bauthor1").value;
257         var title = document.getElementById("btitle").value;
258         var bookformat = document.getElementById("ATformat").value;
259         if(author == "" || title == "" || bookformat == ""){
260             msg1.textContent = "Please fill all the textboxes!"
261             msg1.style.color = "darkred"
262             return false;
263         }else{
264             if (bookformat=='json'){
265                 window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&title="+title+"&bookformat="+bookformat);
266             }else if(bookformat=='xml'){
267                 window.open("http://localhost:8080/bookstore/serverside/bookformat.php?author="+author+"&title="+title+"&bookformat="+bookformat);
268             }
269         }
270     }
271 }
272 
```

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

```

430      <h3 style="text-align: center; font-size: 30px; font-family:sans-serif;">
431          <b>Book format by Title & Publication Year</b>
432      </h3><hr style="margin-bottom: 45px;" />
433      <div class="row">
434          <div class="col-md-4"><input type="text" id="btitle1" placeholder="Enter book title" /></div>
435          <div class="col-md-4"><input type="text" id="byear1" placeholder="Enter book publication year" /></div>
436          <div class="col-md-4">
437              <select name="format" class="txtdrop" id="TYformat">
438                  <option class="txtdrop1" value="">Select format</option>
439                  <option class="txtdrop1" value="json">JSON Format</option>
440                  <option class="txtdrop1" value="xml">XML Format</option>
441              </select>
442          </div>
443      </div>
444      <div class="btn-box">
445          <button onclick="SearchByTitleYearFormat()" >
446              <i class="fa fa-search" aria-hidden="true"></i> SEARCH
447          </button>
448      </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.

```

275      function SearchByTitleYearFormat(){
276          var title = document.getElementById("btitle1").value;
277          var pub_year = document.getElementById("byear1").value;
278          var bookformat = document.getElementById("TYformat").value;
279          if(title == "" || pub_year == "" || bookformat == ""){
280              msg2.textContent = "Please fill all the textboxes!";
281              msg2.style.color = "darkred";
282              return false;
283          }else{
284              if (bookformat=='json'){
285                  window.open("http://localhost:8080/bookstore/serverside/bookformat.php?title="+title+"&pub_year="+pub_year+"&bookformat="+bookformat);
286              }else if(bookformat=='xml'){
287                  window.open("http://localhost:8080/bookstore/serverside/bookformat.php?title="+title+"&pub_year="+pub_year+"&bookformat="+bookformat);
288              }
289          }
290      }
291  }
292 }

```

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:

The screenshot shows the 'Book API' application interface. At the top, there is a navigation bar with links: Home, Add Book, Search Book, Advanced Book, View All Books, Book Format, and a welcome message 'Hello user, Welcome!'. Below the navigation bar is a large image of a stack of books. In the center of the image, the words 'Add Book' are displayed in a colorful, 3D-style font. Below this, the title 'Add book' is centered above a form. The form contains fields for author ('Tom Sawyer' and 'Mark Twain'), year ('1876'), description ('An imaginative and mischievous boy named Tom Sawyer'), language ('English'), ID ('1503215679'), publisher ('American Publishing Company'), genre ('Novel'), and a 'SUBMIT' button. A success message 'Book is successfully added.' is visible at the bottom of the form. At the very bottom of the page, there is a dark blue footer bar with the copyright notice: '© 2022 All Rights Reserved By Beeharry Taajuddin |BSE20AFT-2003_19375|'

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:

The screenshot shows a web application titled "Book API". The top navigation bar includes links for Home, Add Book, Search Book, Advanced Book, View All Books, Book Format, and a greeting "Hello user, Welcome!". A large banner in the background features a library interior with bookshelves and the text "Search book by Title". Below the banner, a search form has the placeholder "Search book by title" and a text input containing "titanic". A green "SEARCH" button is visible. The results section displays a single book entry for "TITANIC" with a movie poster image. The poster shows Leonardo DiCaprio and Kate Winslet. Below the poster, the book's title is listed as "TITANIC" along with its details: Author: Leonardo DiCaprio, Kate Winslet; Year Published: 1997; Language: English; Category: Fantasy; ISBN: 2147483647; Description: Story of the sinking Titanic and the beautiful love story; and Best Seller: Beeharry Taajuddin. At the bottom of the page, a dark footer bar contains the copyright notice: "© 2022 All Rights Reserved By Beeharry Taajuddin [BSE20AFT-7003_19375]".

Status: Pass

Post-Conditions: Book title can be successfully searched.

By author

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

Result:

The screenshot shows a web application titled "Book API" with a dark header bar containing navigation links: Home, Add Book, Search Book, Advanced Book, View All Books, Book Format, and a welcome message "Hello user, Welcome!". The main content area features a large banner with the text "Search book by Author". Below the banner, a search form has the input field "Laurence" and a green "SEARCH" button. A message below the search form states "Total books found: 1". The result is displayed in a card format with the title "NINJA TURTLE", an image of the book cover featuring the Teenage Mutant Ninja Turtles, and the author's name "Laurence Fishburne". Detailed book information is listed: Year Published: 2007, Language: Arab, Category: Fantasy, ISBN: 1416934138, Description: "When the world is threatened by an ancient evil, the four adolescent turtles must reunite", and Best Seller: "Enos Moes". At the bottom of the page, a dark footer bar contains the copyright notice: "© 2022 All Rights Reserved By Beeharry Taajuddin /BSE20AFT-2003_19375/".

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:

The screenshot shows a web application titled "Book API". At the top, there is a navigation bar with links: Home, Add Book, Search Book ▾, Advanced Book ▾, View All Books, Book Format, and a welcome message "Hello user, Welcome!". Below the navigation bar is a large banner with the text "Search book by Publication Year" overlaid on a blurred background of a bookstore interior. The main content area has a heading "Search book by publication year" and a search input field with the value "200". Below the search field is a button labeled "SEARCH". A message "Total books found: 3" is displayed. Three book cards are shown, each with a thumbnail image, the title, author, and a brief description. The first card is for "A BEAUTIFUL MIND" (2001), the second for "CATCH ME" (2007), and the third for "NINJA TURTLE" (2007). At the bottom of the page, a footer bar contains the text "© 2022 All Rights Reserved By Beeharry Tajuddin [BSE2004FT-2003_19375]".

Title	Author	Description
A BEAUTIFUL MIND	Russell Crowe, Ed Harris	Brilliant Mathematician who wins the nobel prize in 1994
CATCH ME	Taj	Based on the real life story of Frank Abagnale Jr.
NINJA TURTLE	Laurence Fishburne	When the world is threatened by an ancient evil, the four adolescent turtles must reunite

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:

The screenshot shows the 'Book API' application interface. At the top, there is a navigation bar with links: Home, Add Book, Search Book ▾, Advanced Book ▾, View All Books, Book Format, and a welcome message 'Hello user, Welcome!'. The main area features a large banner with the text 'Search book by Category' overlaid on an image of a library. Below the banner, a search bar has 'Fantasy' typed into it, and a green 'SEARCH' button is visible. A message 'Total books found: 2' is displayed. Two book cards are shown: 'TITANIC' (Author: Leonardo DiCaprio, Kate Winslet, Year Published: 1997, Language: English, Category: Fantasy, ISBN: 2147483647, Description: Story of the sinking Titanic and the beautiful love story, Best Seller: Beeharry Taajuddin) and 'NINJA TURTLE' (Author: Laurence Fishburne, Year Published: 2007, Language: Arab, Category: Fantasy, ISBN: 1416934138, Description: When the world is threatened by an ancient evil, the four adolescent turtles must reunite, Best Seller: Enoos Moos).

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:

The screenshot shows a web application titled "Book API" with a dark header bar containing links for Home, Add Book, Search Book, Advanced Book, View All Books, Book Format, and a welcome message "Hello user, Welcome!". The main content area features a large banner with the text "Search book by Author & Category" overlaid on an image of books. Below the banner is a search form with two input fields: one containing "Taaj" and another containing "a". A green "SEARCH" button is located below the inputs. The search results section displays a single book entry for "CATCH ME" by Taaj, published in 2000. The book cover art shows two men in suits running. The book details include: Author: Taaj, Year Published: 2000, Language: French, Category: Adventure, ISBN: 213745261, Description: "Based on the real life story of Frank Abagnale Jr.", and Best Seller: Beecharry Taajuddin. At the bottom of the page, a dark footer bar contains the text "© 2022 All Rights Reserved By Beecharry Taajuddin [BSE20AFT-2003_19375]".

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:

The screenshot shows a web application titled "Book API". The top navigation bar includes links for Home, Add Book, Search Book, Advanced Book, View All Books, Book Format, and a greeting message "Hello user, Welcome!". The main header features a large banner with the text "Search book by Author And Publication Year" overlaid on an image of an open book and a row of books. Below the banner is a search form with two input fields: one for "Author" containing "a" and another for "Publication Year" containing "2007". A green "SEARCH" button is positioned between the fields. The search results section displays a single book entry: "NINJA TURTLE" by TMNT. The book cover art shows the four Ninja Turtles. Below the cover, the title "NINJA TURTLE" is displayed, followed by detailed book information: Author: Laurence Fishburne, Year Published: 2007, Language: Arab, Category: Fantasy, ISBN: 1416934138, Description: "When the world is threatened by an ancient evil, the four adolescent turtles must reunite", and Best Seller: Enoos Moos. At the bottom of the page, a dark blue footer bar contains the text "© 2022 All Rights Reserved By Beechary Taajuddin [BSE20AFT-2003_19375]".

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:

Book API

Home Add Book Search Book Advanced Book View All Books Book Format Hello user, Welcome!

Total number of books: 7

TITANIC
Author: Leonardo DiCaprio, Kate Winslet
Year Published: 1997
Language: English
Category: Drama
ISBN: 2147483647
Description: story of the sinking titanic and the beautiful love story

A BEAUTIFUL MIND
Author: Russell Crowe, Ed Harris
Year Published: 2002
Language: English
Category: Drama
ISBN: 4340219069
Description: another mathematician who wins the nobel prize in 1998

CATCH ME
Author: Tom Hanks
Year Published: 2000
Language: French
Category: Drama
ISBN: 2157452821
Description: based on the real life story of frank abagnale Jr.

TMNT
Author: Lawrence Schlesinger
Year Published: 2007
Language: Arch
Category: Fantasy
ISBN: 1416934138
Description: When the world is threatened by an ancient evil, the four adolescent turtles must reunite

THE LITTLE PRINCE
Author: Antoine de Saint-Exupéry
Year Published: 1943
Language: French
Category: Novel
ISBN: 2147483647
Description: a young prince who visits various planets in space

ULYSSES
Author: James Joyce
Year Published: 1922
Language: English
Category: Novel
ISBN: 2147483647
Description: a legendary Greek king of Ithaca and the hero of Homers epic poem the Odyssey

TOM SAWYER
Author: Mark Twain
Year Published: 1876
Language: English
Category: Novel
ISBN: 1503215679
Description: an imaginative and mischievous boy named tom sawyer

© 2022 All Rights Reserved By Beeharry Taajuddin [BEEHARRYTAJUDDIN_19213]

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

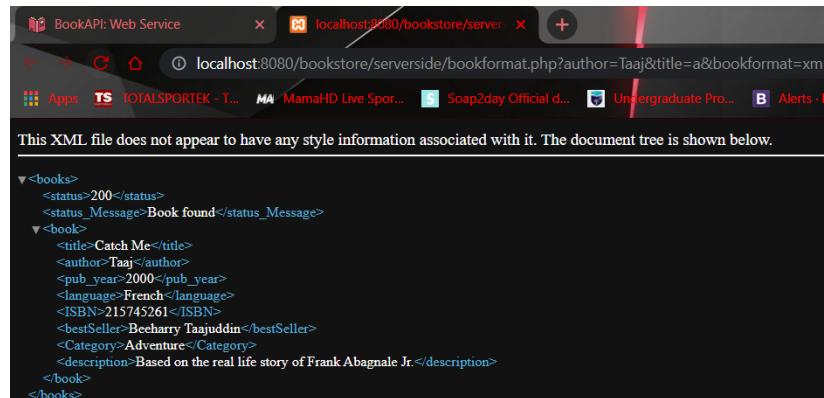


Figure 55 - Testing book format author & title in xml



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

Status: Pass

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