SET09103 Coursework 1 Books Catalogue Development Report

Written by: Piotr Kubicki

Written for: Simon Wells

Date: 24/10/2016

Table of Contents

[Introduction 2](#_Toc465273048)

[Requirements 2](#_Toc465273049)

[Functional 2](#_Toc465273050)

[Non-functional 2](#_Toc465273051)

[Resources and materials required 2](#_Toc465273052)

[Legal requirements 2](#_Toc465273053)

[Design 2](#_Toc465273054)

[Top level use case diagram 3](#_Toc465273055)

[Use case scenarios 3](#_Toc465273056)

[**Use case 1:** User looking for a book by title 3](#_Toc465273057)

[**Use case 2:** User looking for an author books 4](#_Toc465273058)

[**Use case 3:** User rate book 4](#_Toc465273059)

[Back end design 5](#_Toc465273060)

[Database design 5](#_Toc465273061)

[Database design diagram 6](#_Toc465273062)

[Rating calculation algorithm 6](#_Toc465273063)

[Search engine 6](#_Toc465273064)

[Front end design 7](#_Toc465273065)

[Colours 7](#_Toc465273066)

[Colours codes 7](#_Toc465273067)

[Typography 7](#_Toc465273068)

[Sitemap 8](#_Toc465273069)

[Side panel 8](#_Toc465273070)

[Collection view 8](#_Toc465273071)

[Item view 9](#_Toc465273072)

[Tests 9](#_Toc465273073)

[Test cases 9](#_Toc465273074)

[Test results 12](#_Toc465273075)

[Enhancements 14](#_Toc465273076)

[Project evaluation 15](#_Toc465273077)

[Personal evaluation 15](#_Toc465273078)

[References 16](#_Toc465273079)

# Introduction

This document describes development process of books catalogue. Application will be available online and will allow user to search catalogue using different criteria, view book details, show book cover and allow to comment and rate the book. Application should use database to store data. Catalogue should also provide administration panel to allow authorised person to insert new books into collection.

# Requirements

## Functional

Application must:

* show all available books
* allow to select different categories
* allow to select by different authors
* provide search facility
* display book details
* display author details
* allow user to rate book and leave comments under any book
* be available online
* allow admin to add new book
* allow admin to add new author
* allow admin to add new genre

## Non-functional

Application must:

* be finished before deadline
* use appropriate colour scheme
* run in Levinux environment
* be builded using Python Flask
* be intuitive and easy to use

# Resources and materials required

* covers for books
* author portrait photos
* details about books
* details about authors
* machine with working instance of levinux operating system
* icons

# Legal requirements

Application will contain book covers and books authors photography but those will be used only for presentation purpose as a part of university assignment.

All software, frameworks and libraries must be used accordingly to their licenses.

# Design

## Top level use case diagram



## Use case scenarios

### **Use case 1:** User looking for a book by title

**Actor:** User

**Use case overview:** Actor visits page to find some details about Terry Pratchett “Small Gods” book.

**Trigger:** Actor opened page typing base url.

**Precondition 1:** Server is up and running

**Precondition 2:** Actor use machine with internet access

**Basic flow:** Display details of Terry Pratchett “Small Goods” book

**Description:** This scenario describes situation when user have internet access and server is up and running. In this scenario wanted book exists in database. This is a main success scenario.

1. System display books.
2. Actor enter “Equal Rites” into search bar select search by title and click search button.
3. Client send request to server.
4. Server return response with searching results.
5. Client display results to the Actor.
6. Actor select Terry Pratchett “Equal Rites” book.
7. Client send request to the server.
8. Server returns response with book details.
9. Client display book details to the Actor.

**Termination outcome:** Actor close the page.

### **Use case 2:** User looking for an author books

**Actor:** User

**Use case overview:** Actor visit page to find books written by his/her favourite witter.

**Trigger:** Actor opened page typing base URL.

**Precondition 1:** Server is up and running

**Precondition 2:** Actor use machine with internet access

**Basic flow:** Display all books written by selected witter.

**Description:** Scenario describes situation when user have internet access and server is running. Actor looking for a not read book written by his/her favourite author.

1. System display books.
2. Actor select author from page menu to unwind authors panel.
3. Actor select Terry Pratchett from panel.
4. Client send request to the server.
5. Server return response with results to the client.
6. Client display results in alphabetical order by title.

**Termination outcome:** Actor close the page.

### **Use case 3:** User rate book

**Actor:** User

**Use case overview:** Actor visit page to comment book he just read.

**Trigger:** Actor opened page typing base URL.

**Precondition 1:** Server is up and running.

**Precondition 2:** Actor use machine with internet access.

**Basic flow:** Add comment to the selected book.

**Description:** Scenario describes situation when user have internet access and server is running. Actor looking for a book he just read to leave comment and rate them.

1. System display books.
2. Actor use search bar to find book.
3. System display search results.
4. Actor found book and click on them.
5. System display book details.
6. Actor enter username, rate book and enter comment and click ‘Send’ button.
7. System validate user entry and return response
8. System update book rating and display user comment.

**Termination outcome:** Actor close the page.

## Back end design

### Database design

Database will store information about authors, books and ratings with comments. Every author may have many books. Every book may have one to many authors. Every comment must have only one book. Because many to many relations between authors and books tables exists, one more pivot table will be required. This database will allow to get all books related with selected author or authors and vice versa. Developer will also easily find all comments related with selected book. Database will also contain genres table to help order books by their genre. It will be help to generate categories menu panel and may be used to store additional information such as genre description in the future. This prototype application will not contain users table and admin login and password will be hard coded.

### Database design diagram



### Rating calculation algorithm

sr – star ratings

(5 \* 5sr + 4 \* 4sr + 3 \* 3sr + 2 \* 2sr + 1 \* 1sr) / (5sr + 4sr + 3sr + 2sr + 1sr)

Full value equals full star, decimal point value 0.5 up to 0.9 counts as half star.

### Search engine

Search engine will take string of one to many words and will split it into single words. Engine will look for spaces and plus symbols that are used as separators. Every plus (+) symbol will be replaced with “AND” word, and every empty space will be replaced by “OR” word. Resulting string will be used to as database query.

## Front end design

Application will use two main layouts, collection and item layout. Collection layout will be used on the main page, to display search results and other url’s where display of multiple items is required. Item view layout will be used whenever details about the item are required. Administration panel will use same layouts but will contain additional buttons highlighted by red colour. Those buttons will provide additional functionality for administration users.

### Colours

Web application will use monochromatic colour view different shades of blue and white colour for backgrounds and objects, with black and white fonts used where appropriate.

### Colours codes

|  |  |  |
| --- | --- | --- |
| **RGB** | **HEX** |  |
| 0 103 204 | 0067CC |  |
| 41 85 127 | 29557F |  |
| 0 129 255 | 0081FF |  |
| 76 167 255 | 4CA7FF |  |
| 0 64 127 | 00407F |  |
| 217 83 79 | D9534F |  |

### Typography

Application will use sans-serif type fonts for long text to improve readability as this type of the fonts are better for screens. Headings and large characters will use serif type fonts to support visual design. Fonts used here have to be appropriate for wide range of users.

Long text, descriptions: Tahoma

Font-family: Tahoma, Geneva, sans-serif;

Headings: Georgia

Font-family: Georgia, Serif;

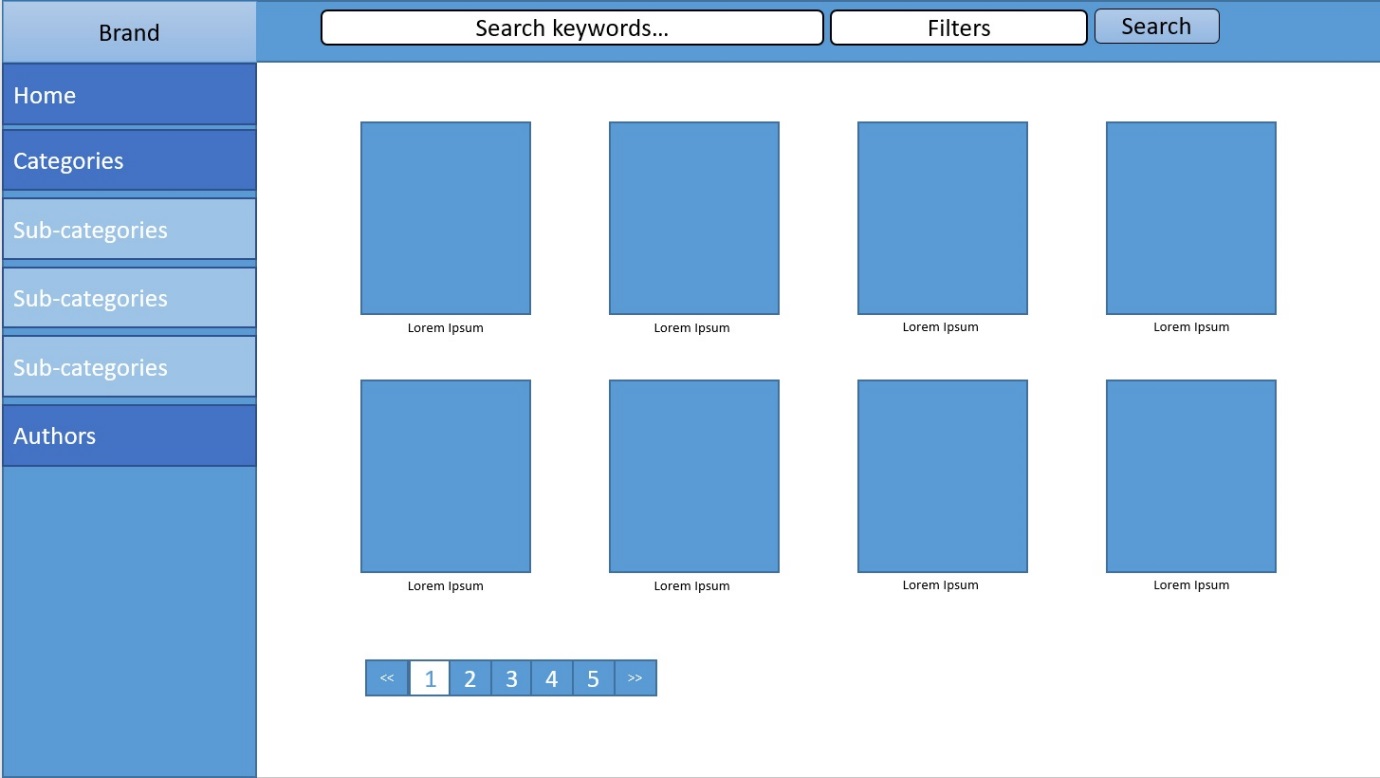
### Sitemap



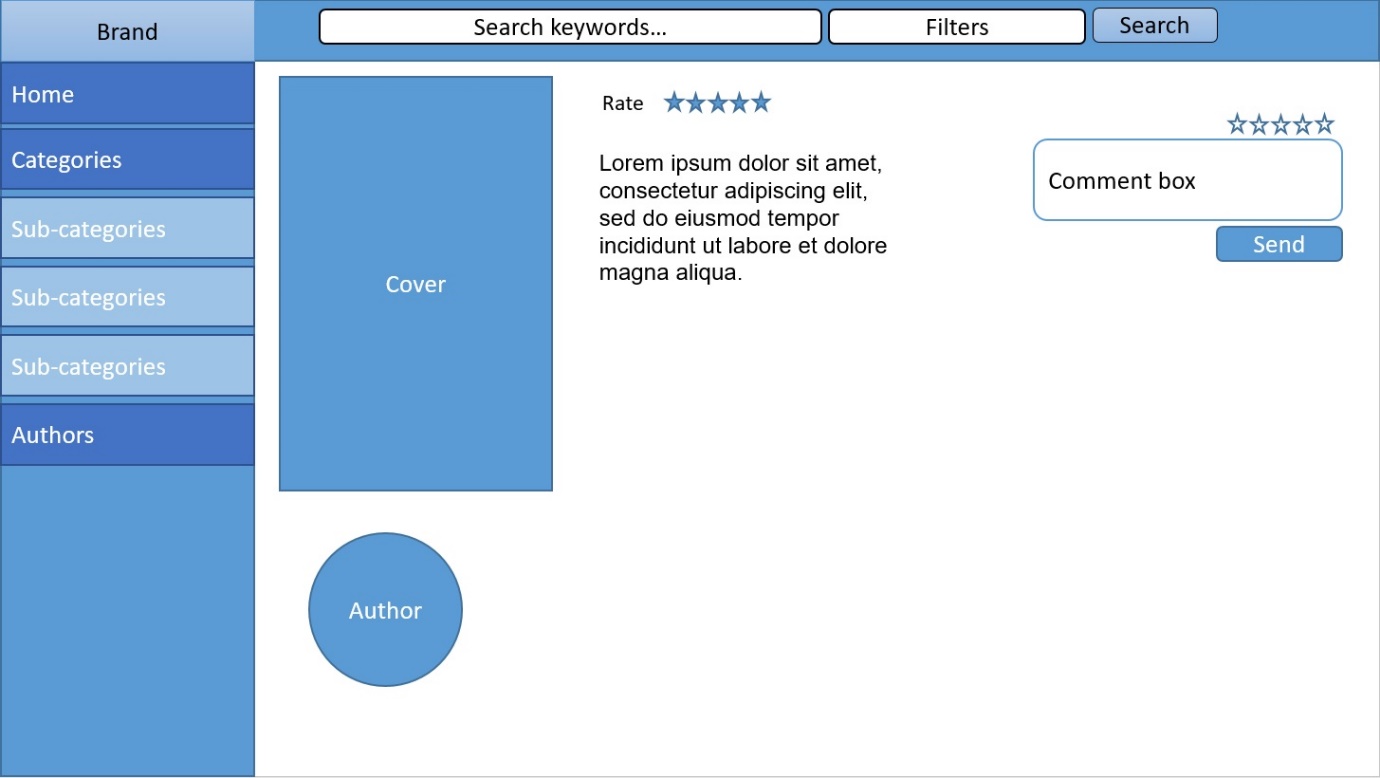
### Side panel

Side panel will be located on the left side of the web app. On very top part of the panel will be located branding section with the app logo. Under branding section user will find menu. Menu will be divided into different sections such as ‘genres’ and ‘authors’, those sections will expand when clicked to show sub categories. When one section expands other sections will collapse.

### Collection view



### Item view



# Tests

## Test cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test case** | **Test data** | **Expected result** | **Actual result** | **Status** | **Actions** |
| 1 | Display collection as normal user | n/a | Collection page displayed |  |  |  |
| 2 | Display detailed page as normal user | n/a | Detailed page displayed |  |  |  |
| 3 | Rate unrated book | Username: Tester  Rating: 3  Comment: Test message | Book rating updated displaying three stars. Comment displayed with correct time and username |  |  |  |
| 4 | Rate previously rated book | Book: same as in test id 3  Username: Tester2  Rating: 4  Comment: Test message2 | Book rating updated displaying 3,5 star  Comment displayed with correct time and username |  |  |  |
| 5 | Display books for selected genre as normal user | n/a | Display only books for selected genre |  |  |  |
| 6 | Display books for selected author as normal user | n/a | Display only books written by selected author |  |  |  |
| 7 | Display home directory as normal user | n/a | Root page displayed |  |  |  |
| 8 | Access admin section before login | url localhost:5000/admin | User redirected to login page |  |  |  |
| 9 | Login to application using incorrect password | url localhost:5000/login  username: admin  password: testpass | User redirect to login page, error message displayed |  |  |  |
| 10 | Login to application using incorrect username | url localhost:5000/login  username: tester  password: password | User redirect to login page, error message displayed |  |  |  |
| 11 | Login to application using correct data | url localhost:5000/login  username: admin  password: password | User redirected to admin collection view |  |  |  |
| 12 | Display admin collection view as admin | url localhost:5000/admin | Collection view displayed with admin functionality |  |  |  |
| 13 | Display book details as admin | n/a | Book details page displayed with admin functionality |  |  |  |
| 14 | Display books for selected category as admin | n/a | Display only books for selected category with additional admin functionality |  |  |  |
| 15 | Display books for selected author as admin | n/a | Display only books for selected author with admin functionality |  |  |  |
| 16 | Create new genre | Name: Test genre | New genre created and displayed |  |  |  |
| 17 | Create new author | First name: Stephen  Last name: Prata  D.O.B: 1967  D.O.D: Null  Photo: any jpg image | New author added and displayed |  |  |  |
| 18 | Create new author with empty first name field | First name: null  Last name: Woods  D.O.B: 1956  D.O.D: 2012  Photo: any jpg image | Error message displayed |  |  |  |
| 19 | Create new book with single author | Title: Test book  Authors: Terry Pratchett  Publisher: test  Year: 2016  Genre: Fiction  Pages: 2  Description: short description  Cover: any jpg image | New book created and displayed |  |  |  |
| 20 | Create new book with multiple authors | Title: Test book 2  Authors: Terry Pratchett, Stephen Prata  Publisher: test  Year: 2016  Genre: Fiction  Pages: 123  Description: short description  Cover: any jpg image | New book created and displayed |  |  |  |
| 21 | Create new book with empty title field | Title: null  Author: Stephen King  Publisher: test  Year: 1976  Genre: Horror  Pages: 10  Description: null  Cover: any jpg image | Error message displayed |  |  |  |

## Test results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test case** | **Test data** | **Expected result** | **Actual result** | **Status** | **Actions** |
| 1 | Display collection as normal user | n/a | Collection page displayed | As expected | Pass | n/a |
| 2 | Display detailed page as normal user | n/a | Detailed page displayed | As expected | Pass | n/a |
| 3 | Rate unrated book | Username: Tester  Rating: 3  Comment: Test message | Book rating updated displaying three stars. Comment displayed with correct time and username | As expected | Pass | n/a |
| 4 | Rate previously rated book | Book: same as in test id 3  Username: Tester2  Rating: 4  Comment: Test message2 | Book rating updated displaying 3,5 star  Comment displayed with correct time and username | As expected | Pass | n/a |
| 5 | Display books for selected genre as normal user | n/a | Display only books for selected genre | As expected | Pass | n/a |
| 6 | Display books for selected author as normal user | n/a | Display only books written by selected author | As expected | Pass | n/a |
| 7 | Display home directory as normal user | n/a | Root page displayed | As expected | Pass | n/a |
| 8 | Access admin section before login | url localhost:5000/admin | User redirected to login page | As expected | Pass | n/a |
| 9 | Login to application using incorrect password | url localhost:5000/login  username: admin  password: testpass | User redirect to login page | As expected | Pass | n/a |
| 10 | Login to application using incorrect username | url localhost:5000/login  username: tester  password: password | User redirect to login page | As expected | Pass | n/a |
| 11 | Login to application using correct data | url localhost:5000/login  username: admin  password: password | User redirected to admin collection view | As expected | Pass | n/a |
| 12 | Display admin collection view as admin | url localhost:5000/admin | Collection view displayed with admin functionality | As expected | Pass | n/a |
| 13 | Display book details as admin | n/a | Book details page displayed with admin functionality | As expected | Pass | n/a |
| 14 | Display books for selected category as admin | n/a | Display only books for selected category with additional admin functionality | As expected | Pass | n/a |
| 15 | Display books for selected author as admin | n/a | Display only books for selected author with admin functionality | As expected | Pass | n/a |
| 16 | Create new genre | Name: Test genre | New genre created and displayed | As expected | Pass | n/a |
| 17 | Create new author | First name: Stephen  Last name: Prata  D.O.B: 1967  D.O.D: Null  Photo: any jpg image | New author added and displayed | As expected | Pass | n/a |
| 18 | Create new author with empty first name field | First name: null  Last name: Woods  D.O.B: 1956  D.O.D: 2012  Photo: any jpg image | Error message displayed | As expected | Pass | n/a |
| 19 | Create new book with single author | Title: Test book  Authors: Terry Pratchett  Publisher: test  Year: 2016  Genre: Fiction  Pages: 2  Description: short description  Cover: any jpg image | New book created and displayed | As expected | Pass | n/a |
| 20 | Create new book with multiple authors | Title: Test book 2  Authors: Terry Pratchett, Stephen Prata  Publisher: test  Year: 2016  Genre: Fiction  Pages: 123  Description: short description  Cover: any jpg image | New book created and displayed | As expected | Pass | n/a |
| 21 | Create new book with empty title field | Title: null  Author: Stephen King  Publisher: test  Year: 1976  Genre: Horror  Pages: 10  Description: null  Cover: any jpg image | Error message displayed | As expected | Pass | n/a |

# Enhancements

In the future application could provide login facility for normal users. That would allow to keep track of user individual preferences. Login to application by Facebook and other social media software would allow user to share favourite books and recommendations with their friends.

Better responsive design would allow application to be used on wider range of devices. At the moment application only supports bigger screens but in the future it could support mobile devices to improve accessibility.

Detailed item view could display links to related books. That could help user to find interesting books quicker.

Improve accessibility by implementing support for screen readers and others accessibility devices

# Project evaluation

The main purpose of this application is to provide users with information about books and their content. Program allow user to specify books genre, title or author to display more specific results.

Application can be used by wide range of users on different age and level of computer literacy. Because of that, it should be intuitive in use to allow user interact with their functions naturally. GUI design helps to achieve that by separating each piece of functionality by different colours and shapes. Those colours stay consistent through entire application that allow user easily recognise purpose of each element. From the other side, application not provide much facility for users with vision disabilities. Program should also provide more support for accessibility devices such as screen readers

Program not display too many elements at the same time to not overwhelm user and keep each element in reasonable size. Application display maximum of 10 books per page witch not only improve readability but also speed up page login by elimination of potentially unnecessary elements. When collection view is well design in terms of number of elements displayed and use of white space, detailed item view should provide more information about presented item.

Most of the code that provide back end functionality is reusable and stored in separate files. That in combination with proper functions names helps to improve readability of the code, and make application easier to maintain in the future. In most cases, well-structured code make program easier to extend with additional functionality. That has been used when admin section that was not planed at the beginning, was added to the application. New functionality was introduced without changes to the previous design.

Application contain two almost identical set of routes functions, one for normal users and one for administration users. It has been done to keep correct URL’s hierarchy. However, I would be better to find an solution to overcome this problem in the future.

Even if some parts of the project could be improved, prototypet has been developed successfully as all required functionality has been provided and tested before the deadline.

# Personal evaluation

During the development process I learned how to use Flask micro framework. Although, I already had some experience on working with similar software, it helped me to improve my understanding of how it works and how effectively use it in my projects.

I decided to use SQL database to store my data within application. As I know from my previous experience many database queries can be reused within application. Because of that, I decided to create python classes for every database table excluding pivot table. That allow me to store queries and operations related with them within their classes, and let me reuse them anywhere in my application. The problem I found with my classed was that some of them depends from each other, for example Book class require Author class to use Author getAuthor method when Author class require Book class getBook method. To overcome that problem, I did some research and found that import statements can be moved to the end of each file or to the methods that will use them. This is possible as python don’t check method implementation until it is not called. I decided to move my import statements to the end of files as I didn’t know how many methods will need them and I didn’t want to repeat myself.

At the beginning of the project I found difficulties with running Levinux instance from pendrive on my Linux operating system at home. The problem was that pendrive has to be used on both Windows OS to access my files in the lab, and Linux OS at home. Unfortunately, pendrive formatted to fat32 file system (for windows use) don’t accept Unix files permissions. That prevent me from making files executable and run them. I also tried to work with the copy of Levinux from my internal drive but then I found I problem to connect with webserver run on Levinux. I spend some time on research to overcome that issues but finally I decided to work with Windows as host operating system to not waste more time and focus on the project.

During the development I been exposed on use of Vim text editor. Since then I used to use more sophisticated text editors such as Atom that include lot of useful functionality helpful during development process. However, more I work with Vim then more I like it. I made some internet research to find some useful commands such as :split and :vsplit commands that allows me to split single Vim instance horizontally and vertically to open and work with multiple files. I also show me how to effectively work with text editor in environment without graphic interface available.

I take most of my design decisions before I started to write actual code. That allow me to find and solve most of the problems earlier and save a lot of time I would have to spend to write and change the code.

When I decided to extend my project with administration section, I realise that it will be necessary to upload image files together with some text data. I decided earlier to send this data using AJAX requests and send data in JSON format. Because of that I had to encrypt image files and attach them to JSON object as text string. During the research I found that it is possible to encode files using Base64 encoding schema to get binary data representation as an ASCII string. Another problems comes when I wanted to decode that binary string again into image file. After number of unsuccessful attempts and online researches I found that web browsers can decode that binary strings to display images. On that stage I decided to remove images stored on application server and keep binary strings inside related database tables.

# References

Coursebook - <https://www.dropbox.com/s/k41vw5a49y64nt7/workbook.pdf?dl=1>

Books covers and information

Amazon.com – [www.amazon.co.uk](http://www.amazon.co.uk)

Authors photos and information

Wikipedia – [www.wikipedia.com](http://www.wikipedia.com)

Code references

W3schools.com – [www.w3schools.com](http://www.w3schools.com)

Stackoverflow – [www.stackoverflow.com](http://www.stackoverflow.com)

Flask doc - <http://flask.pocoo.org/docs/0.11/>

Jijna2 doc - <http://jinja.pocoo.org/docs/dev/>

Select input fields

Select2 - <https://select2.github.io/>

JavaScript library

jQuery - <https://jquery.com/>

Graphic Interface Framework

Bootstrap - <http://getbootstrap.com/>