RAYANN TEDDS – 205CDE REFLECTIVE REPORT

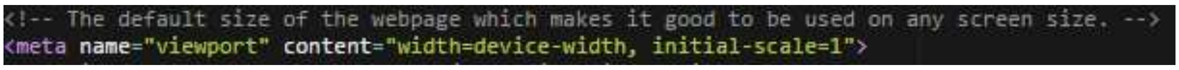
GitHub Repository Link: https://github.com/RayannTedds/bookreviews
YouTube Link: https://youtu.be/F8wt8j0Yji0

MAY 1, 2016

COVENTRY UNIVERSITY
205CDE – Developing the Modern Web

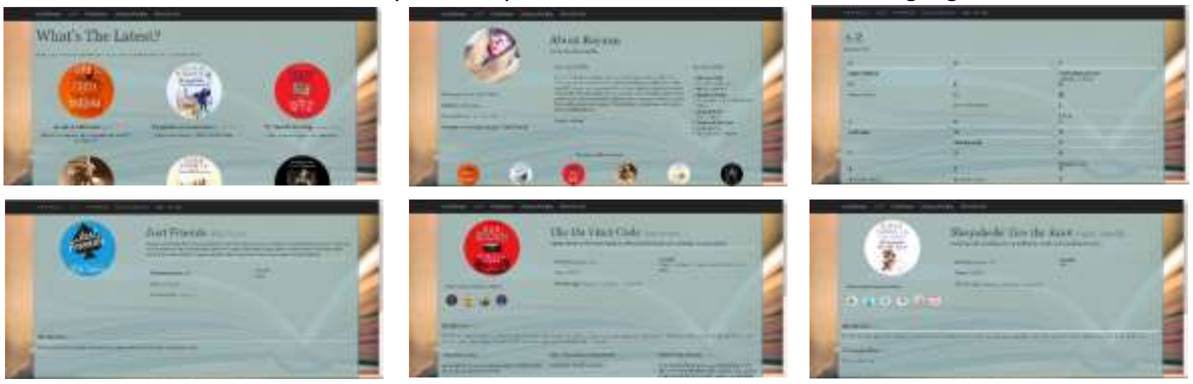
During this module, I have created a website which holds information about books which I have read and given reviews on over time and in this report, I am going to be reflective upon its development. I have used a variety of techniques such as HTML5, CSS, JavaScript, sqlite3 (database), Flask and Bootstrap to create a dynamic and professional website. Many features are present within this website including: the use of an sqlite3 database to populate book review pages; the use of modals to give users an insight into books before having to see the entire review page and a comment/review form (fully validated) which allows users to submit their views on any book of their choice – upon submission of this form, two emails are automatically sent with content to the user and then to the admin of the site (in this case my own personal email address). Finally, for each book mentioned, the user is supplied with links in order to go and purchase the book from different suppliers such as Amazon.

The use of HTML5, CSS and Bootstrap has enabled me to design a website which coincides with the design principles found in professional websites today. I ensured that a colour scheme of blues and creams was used throughout the site and ensured that my designs allowed for alterations for different screen sizes – this ensures that my site can be viewed by the maximum amount of people possible, using a variety of different devices. This was mainly achieved through the following code:

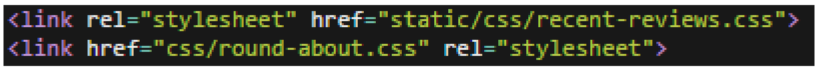


```html
<!-- The default size of the webpage which makes it good to be used on any screen size. -->
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Bootstrap was used in 2 main ways: I found the use of the grid layout to be extremely helpful as it enabled my content to be structured in a uniform and effective way. Below, you can see a few screenshots to highlight this.

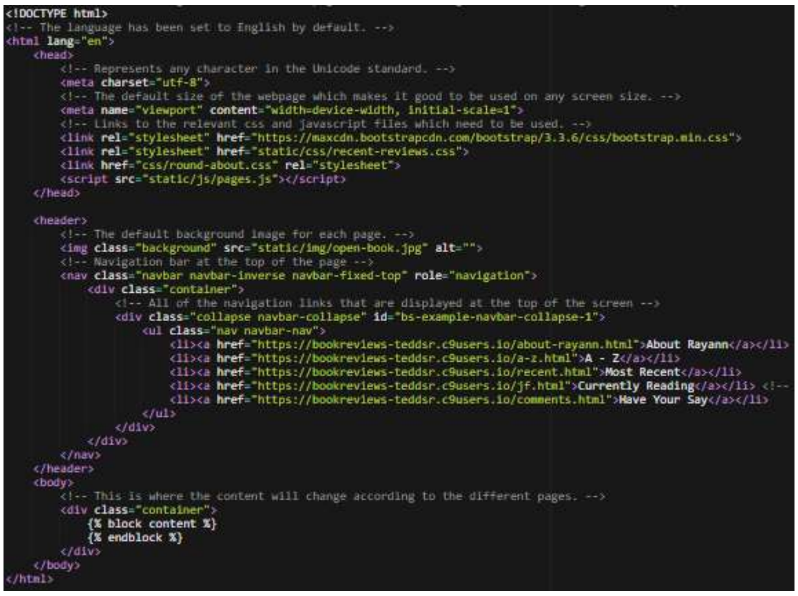


Design principles of a cohesive feel have also been met and this can be seen just by looking at the above screenshots as every page matches each other. This has been achieved through the use of my own CSS code as well as the use of a Bootstrap CSS file called 'round-about'. The look and feel of the website was achieved through the linking of such files with the code shown below:

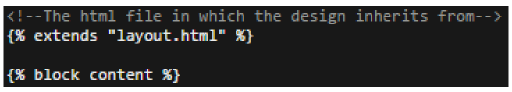


```html
<link rel="stylesheet" href="static/css/recent-reviews.css">
<link href="css/round-about.css" rel="stylesheet">
```

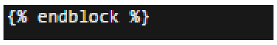
This leads me on nicely to my second use of Bootstrap; through the use of my layout.html file. This code held all the relevant initialisation files and links, as well as the navigation bar for the website itself. Every page of the website (excluding the initial page), then inherits from this file. This ensured that code was not constantly repeated on each page, saving time, effort and a lot of unnecessary coding. On the left, you can see the layout.html code itself and on the right, you can see the code which allows the template to be used on the given webpage. Between 'block content' and 'endblock' is where all the content for the particular page is held.



```html
<!DOCTYPE html>
<!-- The language has been set to English by default. -->
<html lang="en">
    <head>
        <!-- Represents any character in the Unicode standard. -->
        <meta charset="utf-8">
        <!-- The default size of the webpage which makes it good to be used on any screen size. -->
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Links to the relevant css and javascript files which need to be used. -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
        <link rel="stylesheet" href="static/css/recent-reviews.css">
        <link href="css/round-about.css" rel="stylesheet">
        <script src="static/js/pages.js"></script>
    </head>

    <header>
        <!-- The default background image for each page. -->
        <img class="background" src="static/img/open-book.jpg" alt="">
        <!-- Navigation bar at the top of the page -->
        <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
            <div class="container">
                <!-- All of the navigation links that are displayed at the top of the screen -->
                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li><a href="https://bookreviews-teddsr.c9users.io/about-rayann.html">About Rayann</a></li>
                        <li><a href="https://bookreviews-teddsr.c9users.io/a-z.html">A - Z</a></li>
                        <li><a href="https://bookreviews-teddsr.c9users.io/recent.html">Most Recent</a></li>
                        <li><a href="https://bookreviews-teddsr.c9users.io/jf.html">Currently Reading</a></li> <!--
                        <li><a href="https://bookreviews-teddsr.c9users.io/comments.html">Have Your Say</a></li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <body>
        <!-- This is where the content will change according to the different pages. -->
        <div class="container">
            {% block content %}
            {% endblock %}
        </div>
    </body>
</html>
```

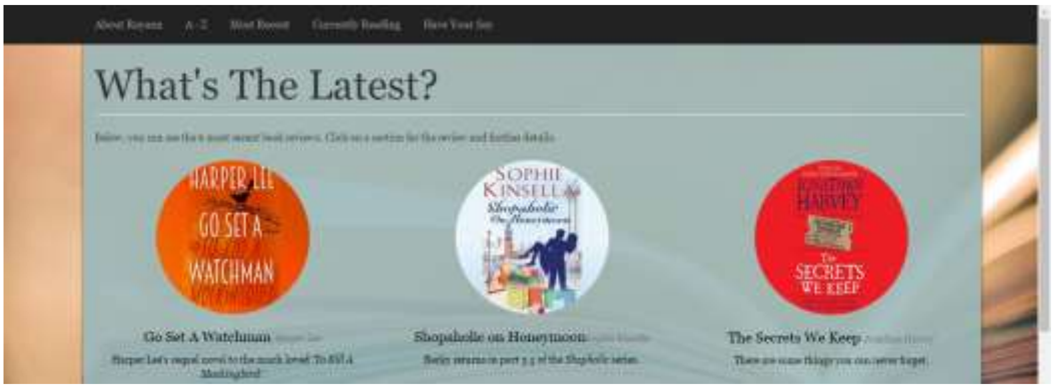


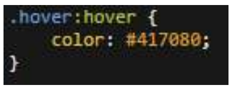
```html
<!--The html file in which the design inherits from-->
{% extends "layout.html" %}

{% block content %}
```



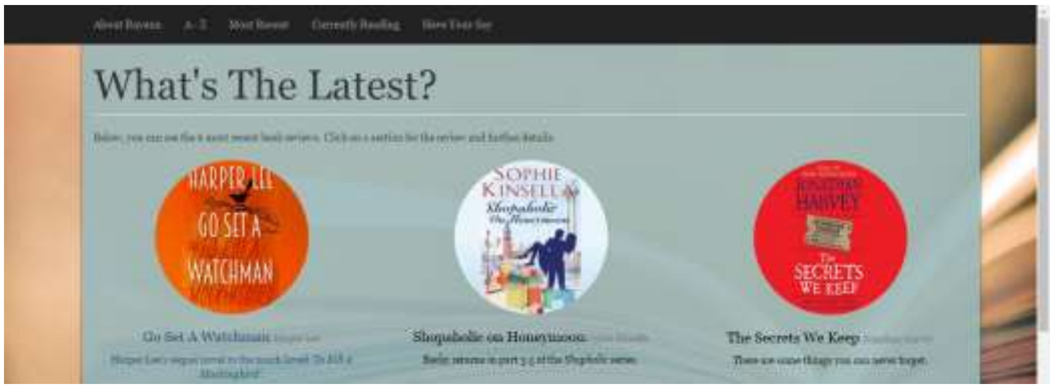
```html
{% endblock %}
```

User interactivity has been active through the use of CSS, HTML and JavaScript also. An example is when a user hovers the mouse over any type of link, then the colour of the text changes, for example, from black to blue – this provides feedback to the users and also helps to keep up with the cohesive feel of the website itself. Screenshots and code snippets of this are shown below:

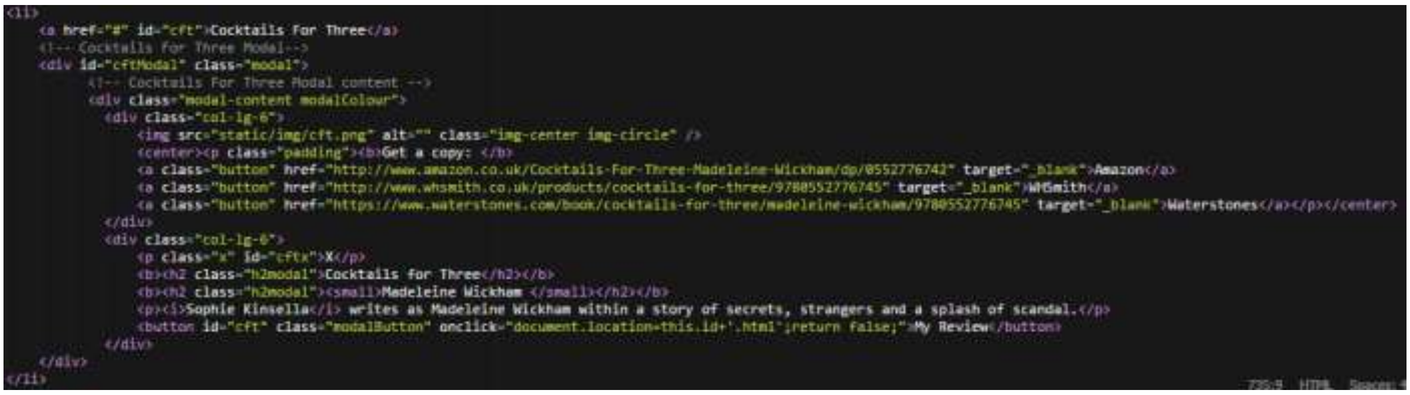




```
.hover:hover {
    color: #417080;
}
```

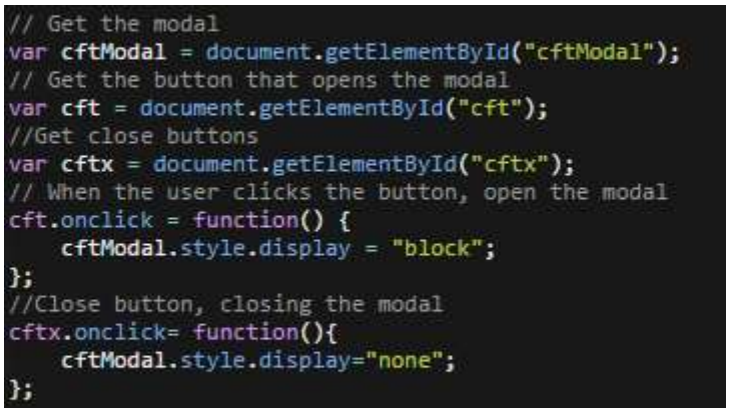


In order to increase my technical ability, I decided to use HTML and JavaScript to create modals which would allow for a book information preview whereby a user could then decide if they wanted to go further to read the reviews or not. This feature is used within the a-z.html file and code snippets and screenshots are shown below to illustrate this feature.

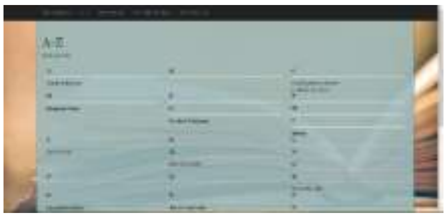


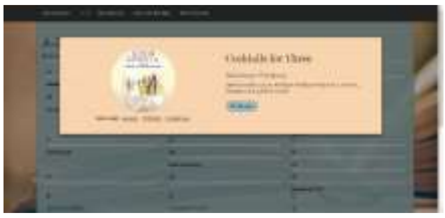
```
<li>
    <a href="#" id="cft">Cocktails For Three</a>
    <!-- Cocktails for Three Modal-->
    <div id="cftModal" class="modal">
        <!-- Cocktails For Three Modal content -->
        <div class="modal-content modalColour">
          <div class="col-lg-6">
              <img src="static/img/cft.png" alt="" class="img-center img-circle" />
              <center><p class="padding"><b>Get a copy: </b>
              <a class="button" href="http://www.amazon.co.uk/Cocktails-For-Three-Madeleine-Wickham/dp/0552776742" target="_blank">Amazon</a>
              <a class="button" href="http://www.whsmith.co.uk/products/cocktails-for-three/9780552776745" target="_blank">WHSmith</a>
              <a class="button" href="https://www.waterstones.com/book/cocktails-for-three/madeleine-wickham/9780552776745" target="_blank">Waterstones</a></p></center>
          </div>
          <div class="col-lg-6">
              <p class="x" id="cftx">X</p>
              <b><h2 class="h2modal">Cocktails for Three</h2></b>
              <b><h2 class="h2modal"><small>Madeleine Wickham </small></h2></b>
              <p><i>Sophie Kinsella</i> writes as Madeleine Wickham within a story of secrets, strangers and a splash of scandal.</p>
              <button id="cft" class="modalButton" onclick="document.location=this.id+'.html';return false;">My Review</button>
          </div>
    </div>
</li>
```

Above, I have defined the modal and its content and below you can see the corresponding JavaScript on the left, with the corresponding results on the right.



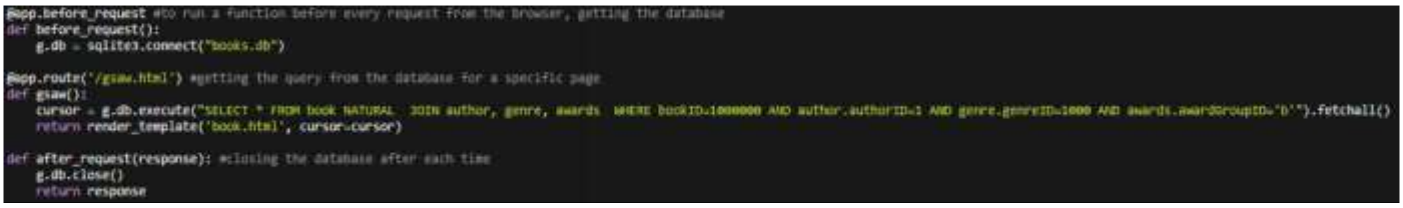
```
// Get the modal
var cftModal = document.getElementById("cftModal");
// Get the button that opens the modal
var cft = document.getElementById("cft");
//Get close buttons
var cftx = document.getElementById("cftx");
// When the user clicks the button, open the modal
cft.onclick = function() {
    cftModal.style.display = "block";
};
//Close button, closing the modal
cftx.onclick= function(){
    cftModal.style.display="none";
};
```





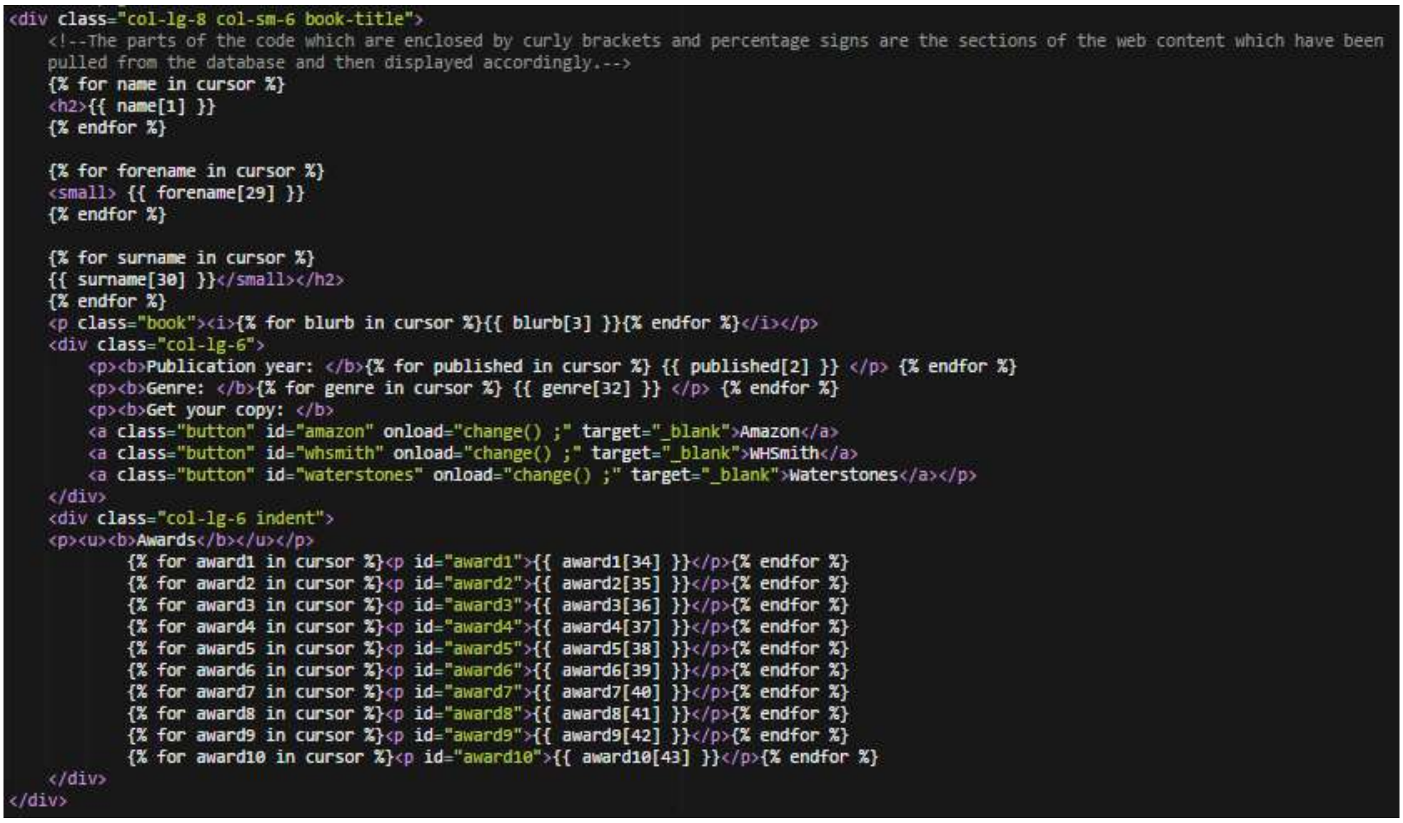
For my website, I heavily relied upon the Flask framework for Python as this allowed for the incorporation of advanced features such as retrieving and using information from a database to routing to comment pages with email features built in. Flask was initially hard to get my head around but once I was familiar with the framework it became apparent that without it, these functions would have been a lot harder to produce and I understood the power Flask has upon web development.

To begin with, I created a sqlite3 database which held all the data and information of the reviewed books from the site. The database table schemas can all be found in Appendix A. Once the database tables were been populated, I then went on to retrieve the required information using Python and Flask in my main routing file. In order to retrieve the desired information, queries are passed each time a new page request is made (a book link is clicked on). A code snippet example of this is shown below:



```python
@app.before_request #to run a function before every request from the browser, getting the database
def before_request():
    g.db = sqlite3.connect("books.db")

@app.route('/gsaw.html') #getting the query from the database for a specific page
def gsaw():
    cursor = g.db.execute("SELECT * FROM book NATURAL  JOIN author, genre, awards  WHERE bookID=1000000 AND author.authorID=1 AND genre.genreID=1000 AND awards.awardGroupID='D'").fetchall()
    return render_template('book.html', cursor=cursor)

def after_request(response): #closing the database after each time
    g.db.close()
    return response
```

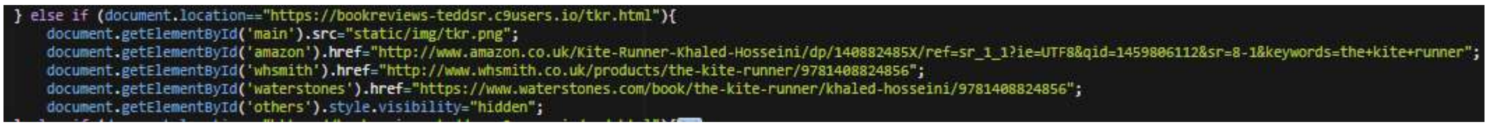
You can see that Flask connects the site to the database, a SQL query is run and then the content is displayed accordingly in the style of the template ‘book.html’. After this has occurred, the database is closed. To actually ensure that the content is displayed on screen, I then ensured that the HTML file had triggers for the system to ensure that the correct database field/record is displayed where it should be. You can see an example of this HTML code below (it is the use of curly brackets and percentage signs):



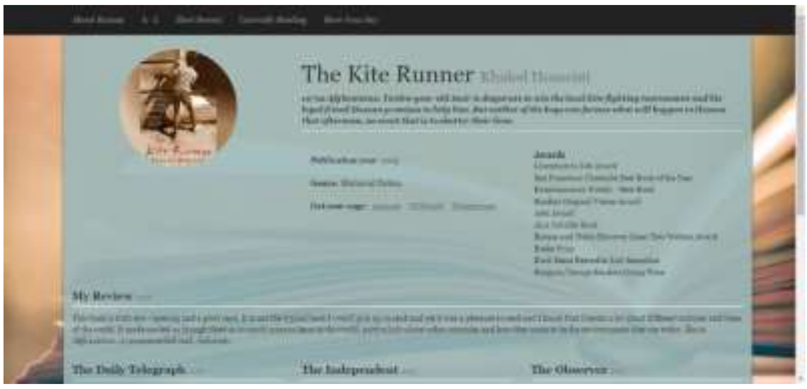
```html
<div class="col-lg-8 col-sm-6 book-title">
    <!--The parts of the code which are enclosed by curly brackets and percentage signs are the sections of the web content which have been
    pulled from the database and then displayed accordingly.-->
    {% for name in cursor %}
    <h2>{{ name[1] }}
    {% endfor %}

    {% for forename in cursor %}
    <small> {{ forename[29] }}
    {% endfor %}

    {% for surname in cursor %}
    {{ surname[30] }}</small></h2>
    {% endfor %}
    <p class="book"><i>{% for blurb in cursor %}{{ blurb[3] }}{% endfor %}</i></p>
    <div class="col-lg-6">
        <p><b>Publication year: </b>{% for published in cursor %} {{ published[2] }} </p> {% endfor %}
        <p><b>Genre: </b>{% for genre in cursor %} {{ genre[32] }} </p> {% endfor %}
        <p><b>Get your copy: </b>
        <a class="button" id="amazon" onload="change() ;" target="_blank">Amazon</a>
        <a class="button" id="whsmith" onload="change() ;" target="_blank">WHSmith</a>
        <a class="button" id="waterstones" onload="change() ;" target="_blank">Waterstones</a></p>
    </div>
    <div class="col-lg-6 indent">
    <p><u><b>Awards</b></u></p>
            {% for award1 in cursor %}<p id="award1">{{ award1[34] }}</p>{% endfor %}
            {% for award2 in cursor %}<p id="award2">{{ award2[35] }}</p>{% endfor %}
            {% for award3 in cursor %}<p id="award3">{{ award3[36] }}</p>{% endfor %}
            {% for award4 in cursor %}<p id="award4">{{ award4[37] }}</p>{% endfor %}
            {% for award5 in cursor %}<p id="award5">{{ award5[38] }}</p>{% endfor %}
            {% for award6 in cursor %}<p id="award6">{{ award6[39] }}</p>{% endfor %}
            {% for award7 in cursor %}<p id="award7">{{ award7[40] }}</p>{% endfor %}
            {% for award8 in cursor %}<p id="award8">{{ award8[41] }}</p>{% endfor %}
            {% for award9 in cursor %}<p id="award9">{{ award9[42] }}</p>{% endfor %}
            {% for award10 in cursor %}<p id="award10">{{ award10[43] }}</p>{% endfor %}
    </div>
</div>
```

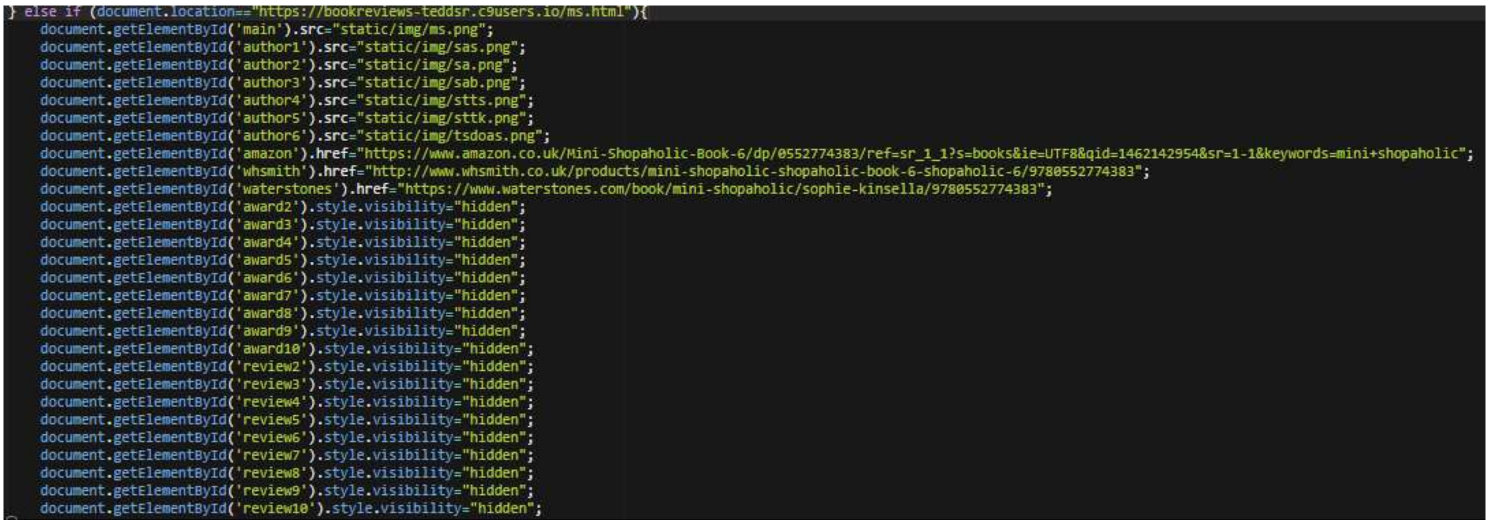
From this, the data is displayed correctly book by book. Examples are shown below.

With the database that I have created, some fields can be null and so this means that each book page can vary as to what content is actually provided – for instance, one book may have 9 awards and 9 reviews whereas another book may have 1 review and no awards. In order to combat this problem, I used JavaScript. Any features that did not need to be shown on screen were simply hidden from view via JavaScript. Below, you can see screenshots and code snippets which highlight its usage.

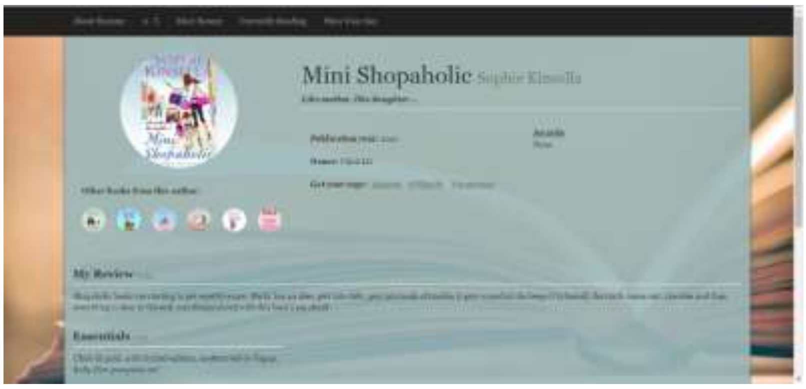


```
} else if (document.location=="https://bookreviews-teddsr.c9users.io/tkr.html"){
    document.getElementById('main').src="static/img/tkr.png";
    document.getElementById('amazon').href="http://www.amazon.co.uk/Kite-Runner-Khaled-Hosseini/dp/140882485X/ref=sr_1_1?ie=UTF8&qid=1459806112&sr=8-1&keywords=the+kite+runner";
    document.getElementById('whsmith').href="http://www.whsmith.co.uk/products/the-kite-runner/9781408824856";
    document.getElementById('waterstones').href="https://www.waterstones.com/book/the-kite-runner/khaled-hosseini/9781408824856";
    document.getElementById('others').style.visibility="hidden";
```





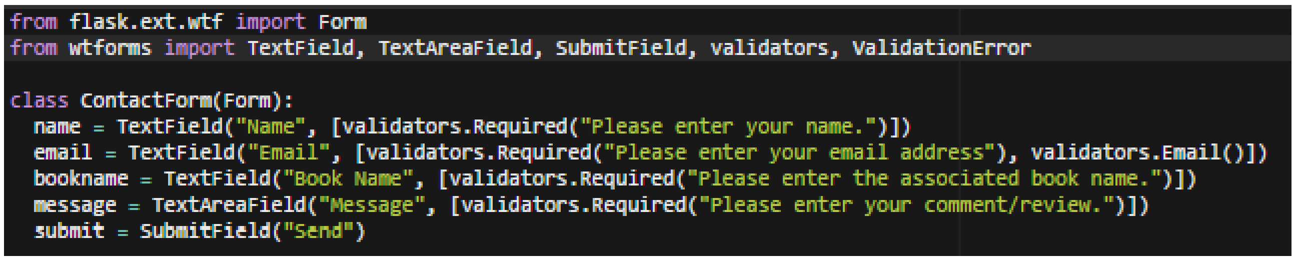
```
} else if (document.location=="https://bookreviews-teddsr.c9users.io/ms.html"){
    document.getElementById('main').src="static/img/ms.png";
    document.getElementById('author1').src="static/img/sas.png";
    document.getElementById('author2').src="static/img/sa.png";
    document.getElementById('author3').src="static/img/sab.png";
    document.getElementById('author4').src="static/img/stts.png";
    document.getElementById('author5').src="static/img/sttk.png";
    document.getElementById('author6').src="static/img/tsdoas.png";
    document.getElementById('amazon').href="https://www.amazon.co.uk/Mini-Shopaholic-Book-6/dp/0552774383/ref=sr_1_1?s=books&ie=UTF8&qid=1462142954&sr=1-1&keywords=mini+shopaholic";
    document.getElementById('whsmith').href="http://www.whsmith.co.uk/products/mini-shopaholic-shopaholic-book-6-shopaholic-6/9780552774383";
    document.getElementById('waterstones').href="https://www.waterstones.com/book/mini-shopaholic/sophie-kinsella/9780552774383";
    document.getElementById('award2').style.visibility="hidden";
    document.getElementById('award3').style.visibility="hidden";
    document.getElementById('award4').style.visibility="hidden";
    document.getElementById('award5').style.visibility="hidden";
    document.getElementById('award6').style.visibility="hidden";
    document.getElementById('award7').style.visibility="hidden";
    document.getElementById('award8').style.visibility="hidden";
    document.getElementById('award9').style.visibility="hidden";
    document.getElementById('award10').style.visibility="hidden";
    document.getElementById('review2').style.visibility="hidden";
    document.getElementById('review3').style.visibility="hidden";
    document.getElementById('review4').style.visibility="hidden";
    document.getElementById('review5').style.visibility="hidden";
    document.getElementById('review6').style.visibility="hidden";
    document.getElementById('review7').style.visibility="hidden";
    document.getElementById('review8').style.visibility="hidden";
    document.getElementById('review9').style.visibility="hidden";
    document.getElementById('review10').style.visibility="hidden";
```


Mini Shopaholic Sophie Kinsella
Awards
None
Other books from this author:
My Review
Essentials

JavaScript has also been used to change the Amazon, WHSmith and Waterstones website links as they vary by book.

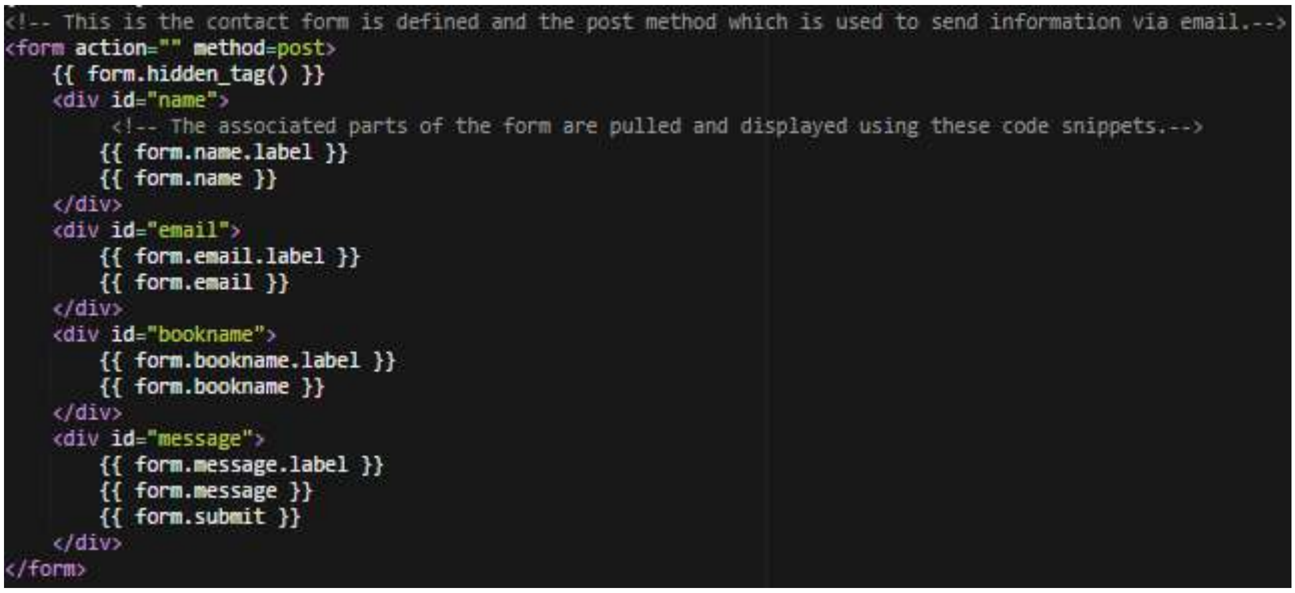
The final fundamental element of my website is the contact form which, upon submission, sends a gratitude email to that user and then an email to the admin of the site (myself) with details of the comments and reviews left by that user. The idea of this is that the admin of the site can then add, easily, any user review they want on their site – this provides a filtering aspect too - any nuisance or offensive comments can be simply ignored by the site owner without the public ever seeing them.

To create such a contact form, a Python file was generated named ‘forms.py’ - this is where the content and validation is actually defined. A code snippet of this can be found on the following page.



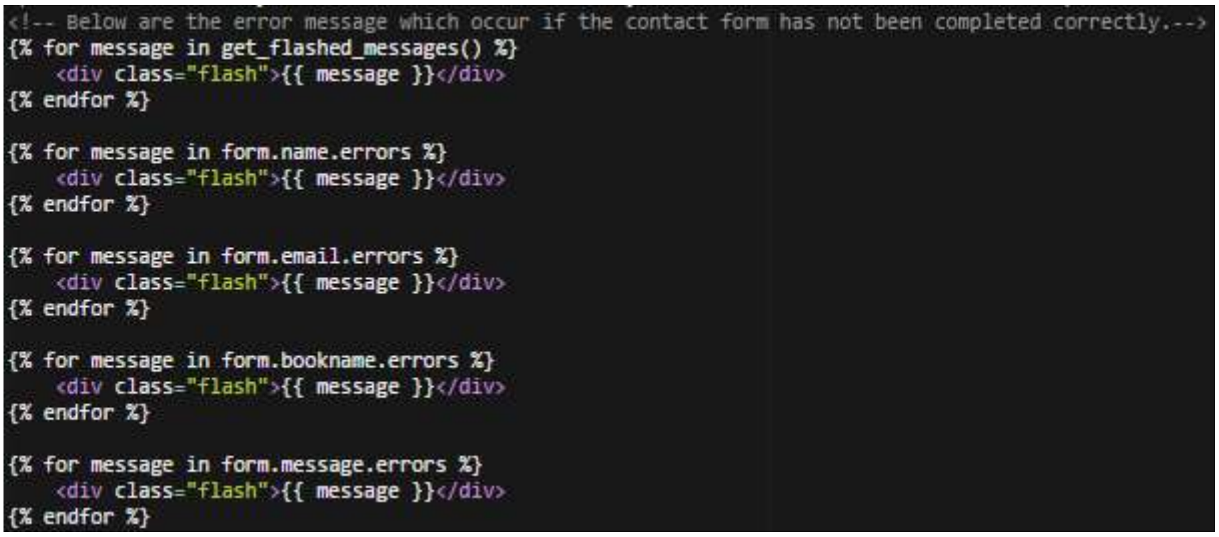
```
from flask.ext.wtf import Form
from wtforms import TextField, TextAreaField, SubmitField, validators, ValidationError

class ContactForm(Form):
  name = TextField("Name", [validators.Required("Please enter your name.")])
  email = TextField("Email", [validators.Required("Please enter your email address"), validators.Email()])
  bookname = TextField("Book Name", [validators.Required("Please enter the associated book name.")])
  message = TextAreaField("Message", [validators.Required("Please enter your comment/review.")])
  submit = SubmitField("Send")
```

Then, like with the database retrieval, the HTML code calls the specific parts of this form in order to be displayed on screen (shown below).



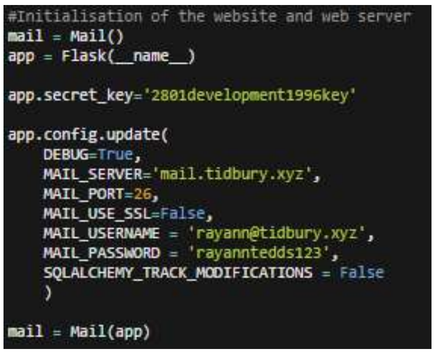
```
<!-- This is the contact form is defined and the post method which is used to send information via email.-->
<form action="" method=post>
    {{ form.hidden_tag() }}
    <div id="name">
        <!-- The associated parts of the form are pulled and displayed using these code snippets.-->
        {{ form.name.label }}
        {{ form.name }}
    </div>
    <div id="email">
        {{ form.email.label }}
        {{ form.email }}
    </div>
    <div id="bookname">
        {{ form.bookname.label }}
        {{ form.bookname }}
    </div>
    <div id="message">
        {{ form.message.label }}
        {{ form.message }}
        {{ form.submit }}
    </div>
</form>
```

Due to the validators in my Python file, I then added HTML code which will display error messages if the form validation is not successful (shown below).

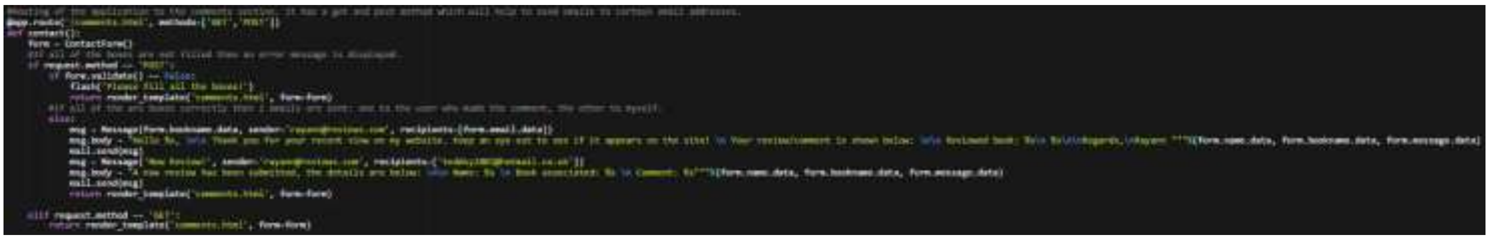


```
<!-- Below are the error message which occur if the contact form has not been completed correctly.-->
{% for message in get_flashed_messages() %}
    <div class="flash">{{ message }}</div>
{% endfor %}

{% for message in form.name.errors %}
    <div class="flash">{{ message }}</div>
{% endfor %}

{% for message in form.email.errors %}
    <div class="flash">{{ message }}</div>
{% endfor %}

{% for message in form.bookname.errors %}
    <div class="flash">{{ message }}</div>
{% endfor %}

{% for message in form.message.errors %}
    <div class="flash">{{ message }}</div>
{% endfor %}
```

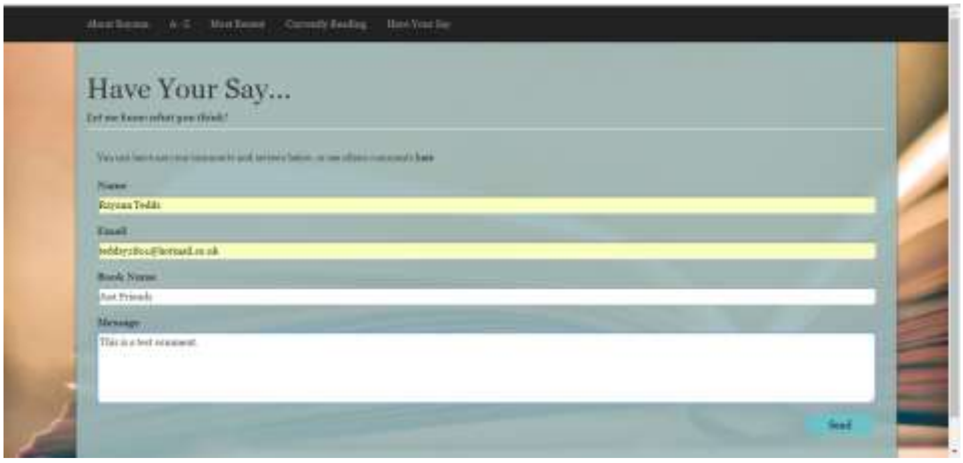
The static form is now in place and so Flask was incorporated to add the functionality and desired effects. Initially, a mail server needed to be set up in order for this function to actually work. The code for this is shown to the left. The use of methods and if…else statements allowed emails to be sent via the website. I have been able to send custom messages for each email which includes the content that the user entered themselves. The code can be found on the following page, along with the proof of emails.



```
#Initialisation of the website and web server
mail = Mail()
app = Flask(__name__)

app.secret_key='2801development1996key'

app.config.update(
    DEBUG=True,
    MAIL_SERVER='mail.tidbury.xyz',
    MAIL_PORT=26,
    MAIL_USE_SSL=False,
    MAIL_USERNAME = 'rayann@tidbury.xyz',
    MAIL_PASSWORD = 'rayanntedds123',
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    )

mail = Mail(app)
```



```
#Routing of the application to the comments section. It has a get and post method which will help to send emails to certain email addresses.
@app.route('/comments.html', methods=['GET','POST'])
def contact():
    form = ContactForm()
    #If all of the boxes are not filled then an error message is displayed.
    if request.method == 'POST':
        if form.validate() == False:
            flash('Please fill all the boxes!')
            return render_template('comments.html', form=form)
        #If all of the are boxes correctly then 2 emails are sent: one to the user who made the comment, the other to myself.
        else:
            msg = Message(form.bookname.data, sender='rayann@reviews.com', recipients=[form.email.data])
            msg.body = """Hello %s, \n\n Thank you for your recent view on my website. Keep an eye out to see if it appears on the site! \n Your review/comment is shown below: \n\n Reviewed book: %s \n %s \n\nRegards,\nRayann """%(form.name.data, form.bookname.data, form.message.data)
            mail.send(msg)
            msg = Message('New Review!', sender='rayann@reviews.com', recipients=['teddsy2801@hotmail.co.uk'])
            msg.body = """A new review has been submitted, the details are below: \n\n Name: %s \n Book associated: %s \n Comment: %s"""%(form.name.data, form.bookname.data, form.message.data)
            mail.send(msg)
            return render_template('comments.html', form=form)

    elif request.method == 'GET':
        return render_template('comments.html', form=form)
```



Have Your Say...

Let me know what you think!

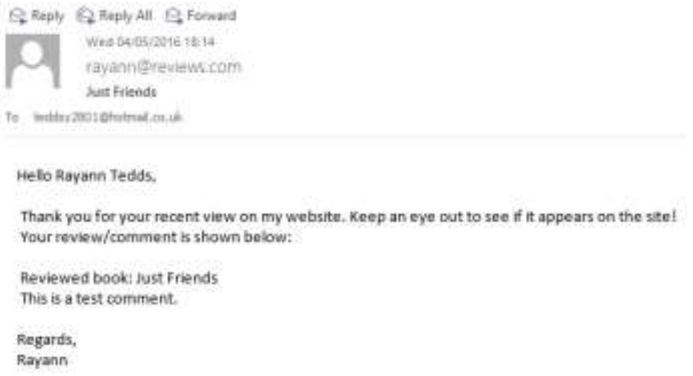
You can leave your comments and reviews below, or see others comments here

Name
Rayann Tedds

Email
teddsy2801@hotmail.co.uk

Book Name
Just Friends

Message
This is a test comment.



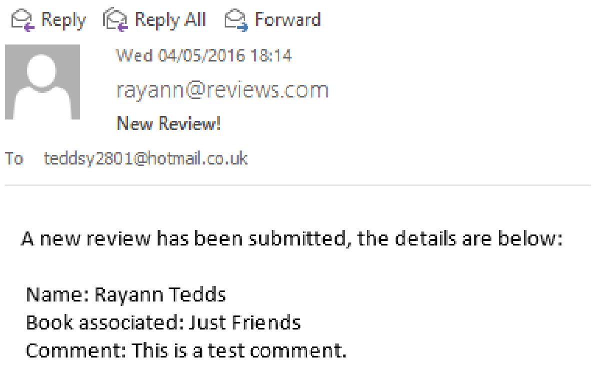
Wed 04/05/2016 18:14
rayann@reviews.com
Just Friends
To teddsy2801@hotmail.co.uk

Hello Rayann Tedds,

Thank you for your recent view on my website. Keep an eye out to see if it appears on the site!
Your review/comment is shown below:

Reviewed book: Just Friends
This is a test comment.

Regards,
Rayann



Wed 04/05/2016 18:14
rayann@reviews.com
New Review!
To teddsy2801@hotmail.co.uk

A new review has been submitted, the details are below:

Name: Rayann Tedds
Book associated: Just Friends
Comment: This is a test comment.

If I was to make any further improvements in the future, I would investigate to see if there was a way in which the book entries into the database could be timestamped so that the recent review page, and aspects throughout the site, are automatically updated. At the moment, the admin of the site will have to change all the references and information within the HTML files when another book has been completed and reviewed. In addition to this, if I wanted to take the site much further, I could incorporate a login for users to register to the site to keep track of all the reviews they have submitted on to the site, rather than just using the comment form and then not being able to retrieve the information they sent in.

To conclude, I am very proud of what I have been able to achieve within this module and I believe my skills have been enhanced massively particularly with the understanding of Flask and web development. My technical skills have, without a doubt, improved and I have shown this through the creation of my website.

Bibliography

Flask, S. (2016) *Sending Email Using Flask* [online] available from <http://stackoverflow.com/questions/19615166/sending-email-using-flask> [1 May 2016]

Grinberg, M. (2014) *Flask Web Development.* Sebastopol, CA: O'Reilly Media [1 May 2016]

Kraynak, J. (2011) *The Complete Idiot's Guide To HTML5 And CSS3*. Indianapolis, IN: Alpha [1 May 2016]

Polepeddi, L. (2013) *An Introduction To Python's Flask Framework* [online] available from <http://code.tutsplus.com/tutorials/an-introduction-to-pythons-flask-framework--net-28822> [1 May 2016]

Polepeddi, L. (2013) *Intro To Flask: Adding A Contact Page* [online] available from <http://code.tutsplus.com/tutorials/intro-to-flask-adding-a-contact-page--net-28982> [1 May 2016]

Unknown (2016) *Round About - Start Bootstrap Template* [online] available from <http://blackrockdigital.github.io/startbootstrap-round-about/#> [1 May 2016]

Thau., (2006) *The Book Of Javascript*. San Francisco: No Starch Press [1 May 2016]

Vander Veer, E. (1997) *Javascript For Dummies Quick Reference*. Foster City, CA: IDG Books Worldwide [1 May 2016]

Walter, S. and Weiss, A. (1996) *The Complete Idiot's Guide To Javascript*. Indianapolis, IN: Que Corp. [1 May 2016]

Wempen, F. (2011) *HTML5*. [Redmond, Wash.]: Microsoft [1 May 2016]

Zakas, N. (2009) *Professional Javascript For Web Developers*. Indianapolis, IN: Wiley Pub. [1 May 2016]

Appendix A – Database Schemas

Book Table Schema

```
CREATE TABLE book (
bookID INT NOT NULL PRIMARY KEY,
name VARCHAR(200) NOT NULL,
published DATE NOT NULL,
blurb TEXT NOT NULL,
authorID INT NOT NULL,
genreID INT NOT NULL,
awardGroupID CHAR(2) NOT NULL,
myReview TEXT NOT NULL,
myReviewDate DATE NOT NULL,
review1 TEXT,
review1author VARCHAR(150),
review2 TEXT,
review2author VARCHAR(150),
review3 TEXT,
review3author VARCHAR(150),
review4 TEXT,
review4author VARCHAR(150),
review5 TEXT,
review5author VARCHAR(150),
review6 TEXT,
review6author VARCHAR(150),
review7 TEXT,
review7author VARCHAR(150),
review8 TEXT,
review8author VARCHAR(150),
review9 TEXT,
review9author VARCHAR(150),
review10 TEXT,
review10author VARCHAR(150),
FOREIGN KEY(authorID) REFERENCES author(authorID),
FOREIGN KEY(genreID) REFERENCES genre(genreID),
FOREIGN KEY(awardGroupID) REFERENCES awards(awardGroupID));
```

Author Table Schema

```
CREATE TABLE author (
authorID INT NOT NULL PRIMARY KEY,
forename VARCHAR(100) NOT NULL,
surname VARCHAR(100) NOT NULL);
```

Genre Table Schema

```
CREATE TABLE genre (
genreID INT NOT NULL PRIMARY KEY,
genre VARCHAR(60) NOT NULL);
```

Award Group Table Schema

```
CREATE TABLE awards (
awardGroupID CHAR(2) NOT NULL PRIMARY KEY,
award1 VARCHAR2(255) NOT NULL,
award2 VARCHAR(255),
award3 VARCHAR(255),
award4 VARCHAR(255),
award5 VARCHAR(255),
award6 VARCHAR(255),
award7 VARCHAR(255),
award8 VARCHAR(255),
award9 VARCHAR(255),
award10 VARCHAR(255));
```