G52GRP Final Group Report

UBOOK - Your Online Library

Group: Parkes1

18th Feb 2016

Group Members:

Ryan Brimble - psyrb1
Oliver Faircliff - psyof
Cait O'Sullivan - psyceo
Yujie Qiu - psyyq1
Tim Sluman - psyts6
Yiming Zhang - psyyz13

Group Supervisor: Andrew Parkes

Description

When you are constantly borrowing textbooks, reference books and fiction books from other people it can be common to forget what you borrowed and from who. It can be even more tricky to know who you've lent books too, all you know is that the book you're looking for was on your bookshelf four months ago, and you might have lent one it the guy in the office three doors down from yours… Our project was to develop a solution for this that would allow people to more easily keep track of the books that they had lent out, and the books that they wanted to borrow from other people. The solution we came up with? UBOOK.

We developed a system that would allow users to log their books as an online library, they would then be able to easily search through their own books and others in the area to find books on a particular subject. Users can add books to their library by taking a picture of the cover and then the system finds the book online, or alternatively, we provided a less high tech but more reliable option for the user to simply type in the name of the book. We have connected to Google books to take advantage of their search algorithms and wealth of knowledge, both in terms of the breadth of their library and the detailed information that they have on each book.

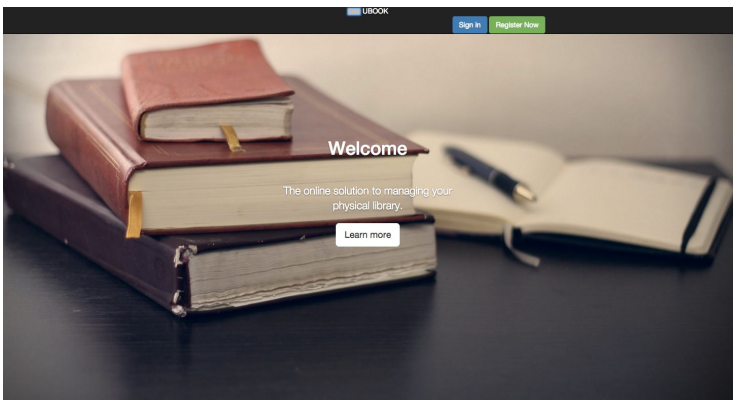
The application would also have a secondary use, it would allow users to search through the libraries of people in their area and request to borrow their books temporarily. It is a system that is designed to let people easily share their own book collection and knowledge and access many more books than they would have had access to otherwise. As the project will be web based the users will be able to share their own library with another user. The user can search for books in both their own libraries and in other people's libraries in order to find the book that they want. The website also keeps track of where the books have gone when they are borrowed and has an alert system via email where users will be able to request books from other people and see the books that they have loaned in order to keep track of books so that they will not get lost or forgotten about.

System Design and User Interface

When designing our system to create and share personal libraries of books we debated about the various ways that we could go about implementing this system. There were pros and cons to various systems, from creating java applications that spoke to a main server (lots of code, difficult to make a smooth and professional UI) to online web applications (have to be careful managing security). In the end we settled on developing a web application, it had relatively few drawbacks and all of the group had some experience of writing web applications.

As we decided to complete the group project as a web application we developed the front end with html, css and Javascript. In particular we chose to utilise bootstrap in order to give our site the polished professional quality that would be expect of us for a year long project. It also has the advantage making the pages responsive and able to display on multiple screen types and shapes and retain its well designed appearance.

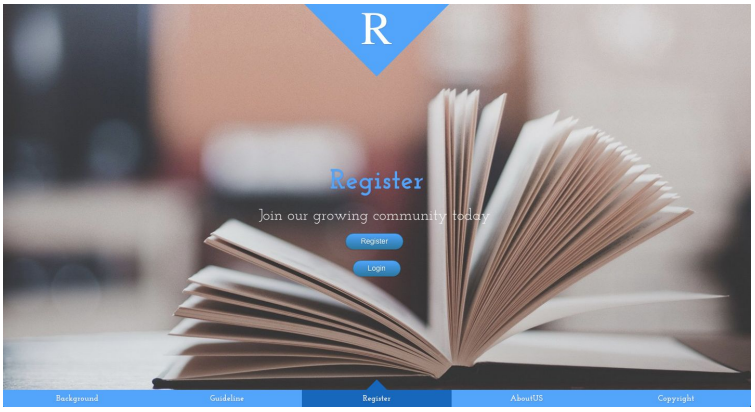
The initial design of the system was created very early in the project and has since gone through several changes. The first page of the website in particular had to be rethought several times as we met limitations that are specified in the report below. However as the front page is the first impression that the user will get of our site we decided in the end to go the route of showing the user clearly what the site embodies, in our case a love of books. Hence we used a picture of a pile of books on the first page background and again variations on this image are used throughout the project. The front page also comes with three key options for any user; to login, to register or to learn more. The first two options, to login and to register, are located in a standard place at the top of the page[Grove, 2010], next to our logo name ‘UBOOK’, they are all located on a plain black bar at the top which makes these stand out to the user. The other button, learn more, takes pride of place in the centre of the screen and its isolation there draws the user’s eye and invites them to find out more information about our site. These design choices were made to give current users a quick way to their own pages (via logging in) and new users a quick way to learn more about the site or choose to register, and they were placed in such a way to separate these two distinct functions of the site and give the first page a minimalistic, professional feel.



Front page of the site.

The user can navigate further through the site by clicking any of these buttons. Both the Register page and the Login page have a design similar to the home page, showing a background of books with a simple and isolated section in the middle for user input. The user simply needs to input an email address and sets a

password to create a new account at our website. User need to enter the registered email address and password to sign in the website. This is in line with our requirements to protect a user's privacy, in part asking for as little personal information as possible, by only asking for a unique login and email (which can be, and are, rolled into one) and a password. All together this help to give the site an overall uniform and united feel.



Learn More page, with focus on 'Register' section.

If instead of logging in or registering straight away the user chooses to select Lean More then the site will jump to a new page which was designed to give the user more information about the website. Unlike the first page, this additional page is divided into five small sections with the headings: 'Background', 'Guideline', 'Register', 'About Us',and 'Copyright'. Again, the background of each section is switched between two pictures of books, to reinforce our theme of books and libraries; the user may switch between the different sections by clicking the name of the section which located in the bottom of the page. The content of each section is clear and concise; for example in the 'About Us' section there are names of our six group members with the roles we played in the team.

After logging in to the site, the user is taken to their personal page, which is a display of the bookshelf page of the individual user. On the top of the bookshelf page the user can use the search bar at the top of the page to enter book information, they can then go about adding this book to their personal library by clicking the 'Add New Book' button. The three other major sections, 'My books', 'Borrowed', and 'On loan' are displayed in the middle of the bookshelf page. The 'My books' section contains all of the books owned by the user. The 'Borrowed' section explicitly lists books currently in the user's library which are borrowed from other users. As for the 'On loan' part, it lists all the books that the user has loaned to other users, and while owned by the user are not currently available in their library. To enhance the visual experience, every book in the bookshelf is displayed as a miniature image of a uniform size. Also, in the bookshelf page, user may click the 'View details' button which provides the users current books as well as past loan and borrowed books. For instance, the details of the 'Borrowed' part may include every book that user have borrowed from other users, and for each borrowed book, it may include the information such as the name of the book, the description of the book, besides, on the right side, it also has two

buttons to allow the user either to extend the permission for the book or return the book immediately.

Should the user wish to add new books to their personal bookshelf, then it is possible to achieve that by clicking on ‘Add New Book’ button. The page will jump to another new page which contains two sections. The first section is named ‘Upload Picture’, here the user can select an image that they have saved, or drag and drop an image into the box, and search for a book based on that image. An OCR reads the text from the image and searches google for a book that matches the words in the search.They also have the option in the other section to ‘take snapshot’, of a book cover, this works by accessing the user’s camera, if they have one, and performs the same OCR extraction and search. The user can always return to the bookshelf page by clicking the home button which locates at the navigator.

To create the desired effect from our design, traditional web programming languages were used to develop the frontend, which includes HTML, CSS and Javascript. The Javascripting is used to create dynamic pages and provide and smooth join between the Javascript on the front end and Node.js on the back end. The basic structure of each page was written in HTML, and the styling of each page is handled by Bootstrap, a free and open-source collection of tools used to help design the pages. Bootstraps standard design splits the page into a twelve column grid, which can make it easier for developers to manage each div class, and also makes it more convenient for batch operation. Using BootStrap also assists in making the site responsive to screen size, something that we needed to take into account for when people want to use the site on their mobiles.

One of the main feature of our website is that it is mobile friendly and that users could use it to take and upload photos from their phones, as per the requirements that we laid out at the beginning of the project. With the help of Bootstrap, our pages can change automatically to fit the different screen size and different devices. Though a completely responsive web design ended with failure (the pictures’ size and structure cannot adjust to the screen size automatically), effort was made to make the website be user-friendly.

One of the initial background images that we chose was of a bookshelf, and while it looked very pretty and fitted with our design pattern we did not fully consider the effect of overlaying the image with text. We quickly found that the text on the spines of the books made our own text hard to read at some points, and the colours of the books and text were had to unify into a single colour scheme. In some cases it just made the content unclear or however in the worst cases the content was obsured completely. Multiple colors are tested, and finally we settled on some simple colors to use in the design, we also swapped out the inappropriate image of bookshelfs for a simpler less complex image, that still fill out design pattern. To make the index page more attractive, we used a wallpaper to fill the background and a navigation bar at the top of the site. For other pages, namely the user’s personal pages, we chose to leave the background blank to give a cleaner more polished look.

To help user get start with our website, another individual page was designed which was pointed from the index page. This page is a brief instruction of how to use the site, and it also introduces our group members. Copyrights is also attached on it. The benefit of this kind of design is the user can easily have a basic understanding of the project. As the instruction page is individual from the project, old users can just ignore the link that points to this page and just log in through the “Login” button.

Implementation and Testing

The Major system components are as follows:

1. Server: this processes the various requests from multiple clients, including requests for webpages; it also processes and search queries that come through and provides a buffer between the client and the database for security purposes.
    a. Login System: performs security checks and logs users in the database in order to track activity and usage on the site.
2. Database: designed in tandem with the site in order to efficiently hold both user information and library information, such that no unnecessary or duplicate information would be held and bloat the database.
3. Frontend: designed to give a professional feel to the site and acts responsively, fulfilling the requirements to make the site work across multiple platforms.
    a. Search Feature: allows the users to have full access to books stored in Google Books and log them as being part of the user's personal library.
    b. Image Recognition Feature: allows the user to take picture of a book cover, or submit a saved image to the site in order to read the text on the cover and perform a search based on that text.

Server

The server is the central component to the system, it manages different users and renders unique pages to each of them. It also has to connect all the components together and implement the logic for the site. We used Node.js with the Express framework to build our server.

Node.js is a version of JavaScript which is used to run code on a server. It is commonly used for web servers because it is also used as the client-side language on websites, meaning you can use the same language on the front and back end of your website. This is an advantage over traditional languages because it removes the inconsistencies which can exist between two different languages, such as being able to transfer data in a common format easily, or constructing and parsing URLs in the same format. Traditional back-end languages, such as PHP, are less suited to web apps, and are better for simpler dynamic sites.

Express is a Node.js framework designed for building servers, it provides functions and components for putting together a complex server with simpler and cleaner code than if you were to write the server with just native Node.js functions. The main component to an Express server is the router, this takes all the web GET and POST requests from the users and routes them to different functions which contain the instructions necessary to create the page to send back to the user.

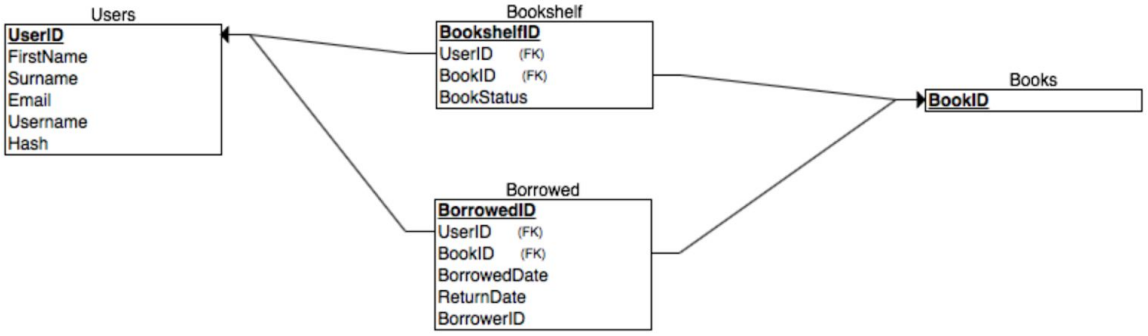
In order to create the page to send to the user's web browsers, the server must take HTML templates, which contain all the HTML markup necessary for a browser to display the page, and insert dynamic content. This includes content specific to each user, like their username and books, and other content which is specific to the individual request, like search results. To do this we used the templating engine Handlebars which plugs into Express.

The server is contained within one class, this class contains methods which setup the server and initialise it. There is one method that initialises the router, this defines the different paths the server should accept, and the functions that the requests should be passed onto, these functions then form the response and send it back. There is a separate module which contains the functions used for connecting to and querying the database, these functions contain the queries to extract or add data to the database. Abstracting all the database queries into a separate module means we do not have to worry about the exact structure of the database in the main server code, we can just call the relevant functions.

Database

The database was designed with speed and efficiency in mind. This meant keeping the amount of columns to a minimum and only storing data which was absolutely necessary. It also meant only searching for ID's rather than strings (where possible) as there are fewer characters that need to match. The first database we designed stored data (author etc.) about books on our local database to aid searches. This initially seemed like a good idea however after testing this approach we decided it would be better to just search the Google Books API for book data and then gather whatever information we needed from there, as trying to contain all the information on each book would bloat the database. If we were storing tags, descriptions, author names and image URL's on our database then it would soon become rather large and would affect the performance. The way we decided to overcome this issue was to only store ID's in the Books table. The ID's wouldn't just be the usual integer values that increase every time a book is added like in most tables (UserID for example) but would actually be the ID of the book on Google Books' API. This way we could simply request that book data from Google. This meant we would still get all the information that our initial database design provided but would only need to have one column in the Books table. It also meant the database would be drastically smaller when lots of books were added. Once Ryan Brimble had designed the database to work in this way it was simply a matter of testing it to ensure that it could be used as intended.

The database design is below:



Testing was carried out locally on Ryan Brimbles mac, the tests included populating the database with dummy data. He then ran the queries similar to those that we would be using on the website to test results and see what issues arose. Fortunately when carrying out these tests no big issues were identified. However one of the minor bug that was revealed was through the choice of data types for dates. By storing the date as an int rather than as a datetime type a function that calculated the return date that a loaned book would be lent out for. Once this was corrected though there appeared to be no other issues in testing.

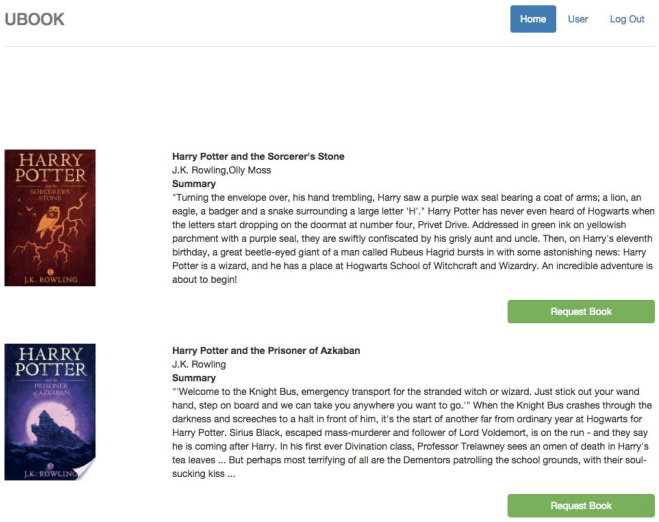
Login System

The login system manages logging in, logging out, and maintaining the login session of different users. It must be secure and robust, otherwise users might not be able to access their account, or their private data could be exposed to other users. We used Passport to create the login system. Passport is a module which was built to work with Express, it provides functions which link up to Express's router to add login functionality. This allows you to ensure only logged-in users can access pages that should only be accessible to logged in users and show the user their unique pages with information about their bookshelf. The Passport module also maintains the user's session on the server, and creates and validates the cookies in the user's browser. The server's session is used to remember users, so they can login once and then the server remembers them so the user doesn't have to enter their login details every time they want to load a new page. The cookies in the browser work in tandem with the session to identify the different users within their session, so the server knows who you are on subsequent requests after logging in.

Search Feature

We decided to implement the search feature of the site by using the Google Books API[Google, 2015] as it offers a large collection of books and information on them as well as an efficient way to access them. As the concept of our site was built around the idea of incorporating

and exploiting the API it was imperative to the project that this feature worked correctly. We were also able to design the database around the information returned by the API in such a way as to make the rest of the program more efficient.



Example of the search feature interfacing with Google Books API

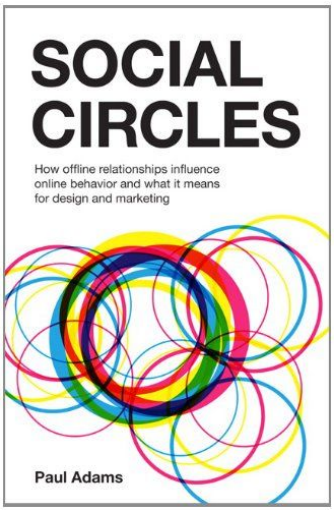
We took steps with the search feature to make it run as quickly as possible, for the convenience of the user and to meet specifications. When performing a search using the Google Books API the data on a book is returned as a list of book objects, these book objects are then parsed from JSON into a readable format that is displayed to the user. The user is only presented with a limited amount of the information that is actually provided by the API; including the title, author and a thumbnail of the book cover. Other information that is returned with the book object, such as a unique identifying number of the book, is used by the database in order to save the book for later reference if the user indicates that they wish to add it to their library. The parsing and handling of the book data is done of the Server side of the site, with the Client only handling the reduced data set in order to improve running speed for the the client.

As we designed the database alongside incorporating this feature we were able to create the database such that a minimum of information could be stored. We elected to err on the side of a smaller database as a smaller database benefits our users because in means that search times are reduced and the site will load quicker for them. A smaller database also means that the jobs of navigating it and maintaining the database are simpler for us a developers. The alternative, of a larger database to hold all the information on each book would only benefit the user in the case that they had a cached version of the website and were using it offline, however as this is not part of our requirement specification a smaller database is more beneficial to the majority of the users. As such the database only contains a unique identifying number of each book, the remaining information on the books is searched for at run time if the user requires it. This is a payoff between system run time for the client and database size. The incorporation of the Google Books API allows the database size to be reduced benefitting the user and us as developers.

OCR

One of the desirable features of our product was for the user to be able to take a picture of the cover of a book and then for that book to be stored on your library. As this would require image recognition software or OCR (Optical Character Recognition) we researched the

options that were available to us in the limited time frame. Though there are lots of pre-build systems for image recognition to detect faces, some even written specifically for node such as Kairos[Kairos, 2015] there were relatively few options for OCR packages. One of the node packages that has been created, image-to-text[Image keywords, 2015], was too generic for our purposes, acting both as an OCR and attempting to find any keywords associated object in the picture, which would give lots of false results in the case of book covers with clear designs on them. In the end we chose to integrate OCR Space[OCR Space, 2016] into our project, which had the benefit of an API and live demo page for easy testing and demonstration. We tested a few book covers using this page before deciding to use this API and the results were promising.



Example of OCR Space Results
Extracted Text:
SOCIAL
CIRCLES
How Offline relationships influence
online behavior and what it means
for design and marketing
O
Paul Adams

As with all image recognition software the accuracy is not 100% and some books with very extravagant covers or fonts caused detection errors. For example, book covers with lots of shapes (especially circles) tend to give inaccurate results as the OCR API tends to interpret the circles as an “O”. Largely though the results are accurate and would get the correct return from Google Books’ API. This is the most difficult error to handle, because while the program thinks that the image had been read a user can quickly see that is hasn’t been read correctly. Other errors that we could produce were easier to handle, for example when image fails to read completely then it returns an error code. Ryan Brimble and Cait O’Sullivan worked together to get the API working and the handle these errors that can be produced. We knew that accuracy would be an issue at the beginning of the project and so we decided that working with these limitation would be acceptable.

Achievements in relation to the Requirements Specification

In our project we achieved most if not all of what we set out to do at the beginning. We created a site that has a smooth interface, fast functionality and includes a number of different features. We were able to include the majority of the requirements that we set out to achieve at the beginning of the project, such as a secure login system. And we also included extra features that were initially listed as stretch tasks in our project, such as using OCR to perform searches based on images.

A key feature on the site was to add in the ability to search for and store information on books that a person would own. Our early research showed that users would not be as likely to use our site if they had to manually input in by hand all the information on the books that they owned. The alternatives that they left us were to either amass our own information and build up a digital library alongside the site, or to utilise someone else's library. After some research into the various online libraries that exist we decided to go with using an official API for Google Books, as they had one of the most complete online libraries and an API that would work seamlessly with our own application. Once a search is performed on our site we send a through the API for a list of books that are relevant to the search, the results are then displayed on the site along with options to add them to a user's bookshelf.

On our site we were able in implement many features that we had initially listed a stretch tasks. One of the features that we were most pleased to include in our site was the ability to take a photo of the cover of a book and perform a search based on the text that it could see. The task is performed by two separate APIs that we have connected together to work in tandem with each other. The API for OCR.space first reads the text that can be seen on the front cover of the book, this is a task that can be performed nealy instantaniously for small images but can have a slight delay for larger image files. Once the text has been read from the image it is then passed into a search that utilises a Google Books API that searches for matches based on the provided text. The results are then displayed on the site as with a normal keyword search, each with the option to add to the user's library, as specified in the early requirements.

On of the search features that we were not able to implement in the time that we were given was to allow the user to restrict their searches to only books on their shelf. This was due to the incompatibility of the standard search feature, which uses the Google Books API to search the Google Books database, and a specialised search feature that would be needed to only search books that were listed against the user’s unique ID in our database. As most of the information about the books is only stored on Google Books, and not on our database this implementing search of our database and then a restricted search on that subset, which we were unable to implement in the time that we had.

Tied into the restricted search feature was a method that would allow users to search the libraries of others in their area. Though the functionality for this feature exists on the database we were unable to create a function on the GUI, partially due to time constraints and partially due to miscommunication within the team about the design of the database and the final GUI. Though we lack this feature that was desired at the beginning of the project since the majority of the back end work is done for it adding in this feature in later iterations of the project should be relatively simple.

In the original requirements our system was specified to be able to work across multiple platforms and on multiple devices. We achieved this by designing a web application that acts responsively based on different screen sizes. In the end we decided that we could only offer limited support for Internet Explorer users, but for other users on Chrome, Firefox, Opera and Safari the site works as expected and testing revealed no issues on these platforms. As it is a web application we were able to do less work focusing on making it work across platforms, and creating different clients based on which platform was accessing our site and dedicate the time that that freed up into making the design of the site that was simple to use and professional to look at.

We made our website user friendly and attractive, which gives it a smooth and professional feel which is inline with the image that we wanted to achieve. In order to achieve this we made a number of design choices, including using Bootstrap in order to have much greater control of the look of site and the CSS. Throughout the site we have used a consistent colour scheme and font selection, as well as high quality images that are related to our product. Together all these elements combine to give our site a unified and user friendly feel.

One of our key requirements was to protect every user's personal information to the best of our abilities, which we have been very successful in doing. In order to do this each user’s login and password are stored as a hashed value, such that if the site is hacked then every user’s password information is protected. We also only store information that is necessary to uniquely identify each user, namely their username and password. In order to protect the user’s security more the database can only be accessed by a few select members of the team.

The database that we made is set up in such a way that it is fast to navigate by both search algorithms and people. We experimented with the amount of information that needed to be stored in order to balance the size of the database against performing operations at run time to extract information, and ended up storing the information the we did. We used third order relationships where possible in the database to reduce the amount of duplicate data to a minimum, which has the added benefit of reducing the search time of the database. All combined this means that our database is fast and efficient and once it was successfully integrated with the site we were able to achieve some of the other design specifications that we had previously laid out.

We also decided that the site and database should be that the site should have good reliability and high maintainability. This means that the website should not easily crash and bugs should be found and resolved in a quick and efficient manner. We achieved this by sticking to recognised coding conventions[Schafer, 2005] and thoroughly testing the code before deployment. This means that the code that we released will be relatively bug free and for

any bugs that are found by sticking to coding conventions we have made the job a programmers reading through the code slightly easier.

Success of the Project

Throughout the project we met several challenges, some we were able to handle by adjusting our expectation of the project and some we were able to to meet head on. We have learned a lot along the way about each other and working together. Some of the problems that we encountered were technical in nature, such as timeout issues with the various APIs that we used. Other problems that we had were more personable, however by pulling together we were able to coordinate ourselves into a working and effective unit. A combination of good teamwork and quick thinking meant that we were able to work around most of the problems in order to create a final project that was better than even we could have anticipated.

One of the elements that we struggled with was revealed early on in the implementation of the site. In the early designs of the site the pages that we mocked up do not show the cover of the books until they are moused over, instead the users was meant to be displayed the spine of the book, as they might see on their own bookshelf. After researching we soon found that it was very rare for the spine of the book to be stored with the cover[Book Data, Google], this meant that we would have to rethink our entire initial design. We were left with two possibilities, the first being to create our own 'spine cover' which would consist of the name of the book and the author printed sideways on a randomly coloured spine. We quickly dismissed this option as it did not fit with our specification to make the books on the user's bookshelf recognisable or at the very least easy to match to its real life counterpart. Variations on this, such as writing a piece of code to pick up the predominant colour on the cover and using that on the spine seemed like overkill for a simple display and processing the cover of every book would slow down the render time for the page, making the app feel slow and clunky. In the end we decided to simply show the cover of every book and abandon the 'bookshelf' feel of the website. This lines up with other similar applications, such as iBooks[iBooks, 2016] or GoodReads[GoodReads, 2016], so would be a standard solution that would be both quick to render and provide a familiar feel to the user. Though the break away from a 'bookshelf' feel would draw away from the unique feel of our site, given the resources that we had access to we would be unable to implement this to the standard that we wanted, so making this small change to a more standard layout would preserve the professional look of the site and still fit with the requirements laid out at the beginning of the project.

One feature that we felt was vital to our page was a navigation bar at the top of the page, this would allow the user to Login, Log Out and perform other actions related to the site. In our initial design we planned to use breadcrumb navigation, however while this is suitable for websites that follow a strict top-down structure the structure of our site is much closer to that of a closed loop. In this case using breadcrumb navigation could cause chaos and cause the user to get lost very quickly. Thus, instead of using breadcrumb navigation, some simple buttons were added and a single navigation bar was designed, which just allows user to return to the main page and log in/out. By using this design pattern, user won't get confused as there are limited choices and limited links for user to choose. Though the user's freedom is decreased, by turning navigation into a simple return and log in/out menu, the efficiency is largely improved and the user's time is saved.

When implementing the OCR feature into the site there were some discrepancies in the return time of results from the API. Ryan Brimble tested this feature by sending different book covers through the API and analysed the return time. The results of this testing showed that time generally seemed to be below three seconds. Occasionally there would be a discrepancy in the analysis time of the book cover, investigation revealed the intricacy of the image did not have a large impact on the time, however sending very large image files through the API could greatly affect the return time. The turnaround time for large images (1600x2400px) being roughly 5 times slower than smaller files (625x829px). This slow processing time even managed to get a timeout response on a few of the largest files. We decided that in the event of the API returning and time out error code that the user will be given the option to enter the title of the book, since it is a rarity it should not impact the user experience too much. Most files (small to medium sized) tend to get a response within 5 seconds however so we can now successfully implement this full feature and meet the requirements spec.

From a project management perspective we worked well together as a group, this part of the project went relatively smoothly. We were able to act together professionally and considerately and avoided large arguments that would have had a detrimental impact on the project. On the whole every member was able to complete their chosen area of work in the time that they had committed to. This was in part because we had a varied mix of people that were all skilled in different areas, this happened to come together perfectly in our project such that we did not lack for a front end developer, or a back end developer, or so on. There was only one area that all of us had relatively little experience, writing a database, and in this case Ryan Brimble readily stepped up to the challenge and was able to learn how to construct an efficient database and provide some standard search queries to the rest of the team to make searching through the database easy and hassle free. In part the reason that we worked together well as a team was because there was no overly dominant personality to conflict with the other team members. Due to us all acting cordially and professionally the whole team was able to empathise with one another and work well as a unit.

If we had to do this project again we would endeavour to add more feature to the site. In our initial requirements we laid out many features that we wanted the site to have, such as image recognition, limited search functionality and location based searches. While we were able to program in some of these features, such as image recognition based searches, due to time constraints on the project we were unable to implement all of them. We do have back end support for some of the features, such as the location based book search, or excluding books that are currently on loan from a particular search. This means that if given more time these features should be simple to add to the site. However due to improper time management of the project we did not delegate enough time to programming in these features the first time, so in doing this project again it would be beneficial to rethink the time plan from the beginning of the project and adjust our expectations and the division of labour across the project accordingly.

In conclusion we completed the group project to the best of our ability in the time that we were given, with most member of the team able to commit to a large amount of time to the

project. There were some feature that we were not able to add to the site, however the majority of key features that we wanted for the site were able to be added, which shows that we were able to prioritise a working site over optional added extras that would improve user experience. We were also able to work well together on the whole, with no major arguments within the group and all of us able to maintain communication throughout the project. We faced challenges, we were able to face them all and adjust to them as they came, as a result while the final result may look different to the original idea that we had the functionality of the site is on the whole up to what we wanted at the beginning of the project. This means that on the whole our project was a success, we faced challenges that we dealt with and we were able to do this in an adult professional manner without blaming team members or causing the project to suffer for it.

Appendix (535 words)

[Grove, 2010] Ralph F. Grove, 2010, Web Based Application Development

[Google, 2015] Google Books API, November 2015, developers.google.com/books

[Kairos, 2015] Kairos-api Node Package, August 2015, www.npmjs.com/package/kairos-api

[Image keywords, 2015] Image-to-Text Node Package, November 2015, tonicdev.com/npm/image-to-text

[OCR Space, 2016] OCR Space API, 2016, ocr.space

[Schafer, 2005] Steven M. Schafer, 2005, Web Standards Programmer's Reference: HTML, CSS, JavaScript, Perl, Python and PHP

[Book Data, Google] Book Data Object, Google Developers, developers.google.com/books/docs/v1/reference/volumes

[iBooks, 2016] Apple iBooks, 2016, apple.com/uk/ibooks/

[GoodReads, 2016] GoodReads, 2016, goodreads.com

Group Meeting 4th February 2016

Present: Ryan Brimble
Oliver Faircliff
Cait O'Sullivan
Yiming Zhang

Absence: Tim Sluman

Yujie Qiu

Minutes:
Summary of progress including work on the implementation of major features of the site such as the ability to login and the creation of a database.
Motion to investigate the usefulness of storing the ISBN number of a book rather than storing the unique identifier provided by Google.
Discussion on the presentation of the site and the development of the front end that users will interact with.

Group Meeting 11th February

Present: Ryan Brimble
Oliver Faircliff
Cait O'Sullivan
Yujie Qiu
Tim Sluman
Yiming Zhang

Absence: n/a

Minutes:
Summary of progress including implementation of the Google Books API to assist with the search feature of the website.
Motion to assess the information that is stored in the database, and whether we should aim to store more information and have a larger database or reduce the amount of information stored and perform a search using the API for each time a book is needed.

Minutes:
Summary of progress including getting the server up and running and setting up a connection between the gitlab repository and books.oliverfaircliff.com
Motion to investigate the use of image recognition and OCR to read the title of books and incorporate it into our project.

Group Meeting 3rd March

Present: Ryan Brimble
Oliver Faircliff
Cait O'Sullivan
Yujie Qiu
Tim Sluman
Yiming Zhang

Absence: n/a

Minutes:
Summary of progress included discussion on how the merging of front end development and backend development into one project.
Motion to adjust the placement of some of the features on the front end in order to make it a more user friendly experience.
Discussion of problems arising from using OCR, such as sending large image files produces a delay of up to 30 seconds between sending and receiving data.

Group Meeting 10th March

Present: Ryan Brimble
Oliver Faircliff
Cait O'Sullivan
Yujie Qiu
Yiming Zhang

Absence: Tim Sluman

Minutes:
Summary of progress included the merging of the latest features into the project including the OCR feature.
Motion to use real users in order to receive feedback on the development of the project, these users would be a mix of academics, such as students and lecturers.
Discussion about how the site could be utilised to provide a service that would allow users to share and swap other items such as CDs and other physical objects; idea rejected due to time constraints on the project.