Rent A Book

H446-03

Name: Niall Braniff

Candidate number: 3020

Centre name: Sandringham school

Centre number: 17535

Contents

Analysis

Problem Description

Lots of people around the world has read books piling up in the corner somewhere, however they do not want to sell them as they may want to read it again or pass them down to their children. These books would then be forgotten about, or they would be thrown out into a landfill, which there has been an increase in due to the rise of eBooks instead of being reused or recycled. I am going to try and solve this problem by designing and building a website which will easily allow them to loan out these books to people while making money from it. The website will also allow people to rent books out, so that they can read them when on holidays or for any other reason. My website would mean that people will be able to read the books without needing to pay the full price of it or worrying where to keep them once read. It can also temporarily give space for the owner to store other things and still being able to get the book back and read it whenever they want. This would all mean that the books will be getting reused instead of being added to a landfill. The owners will be doing all this while making money from loaning them out. The transactions through my website can be done using PayPal, however my website should allow other forms of transaction in case the users do not have a PayPal account. The website will have a search engine to help make searching the website without great effort.

Stakeholders

There will be a broad range of users using the website. This is because it could be a teenager needing to rent out a physics or computer science book, or they may own a Macbeth book and is looking to make a few pounds by lending it out to other students. It could possibly even be an elderly wanting to read old classics such as Treasure Island or Robinson Crusoe. Book enthusiasts or even historians may want to use it to rent out rare books, like The Duke of Reichstadt (Napoleon 11). However, I believe that the most common user of my website would be adults and the elderly, as young people and teenagers are more likely to be just using eBooks, while they would have most likely kept books over the years. This means that I should use a web based system instead of phone apps or gaming consoles as it is the most commonly used method by my stakeholders. The website would work best as it can be accessed by any device able to connect to google, and although a lot of people may not have a smartphone, there is a bigger chance that they own a laptop/ computer or are able to gain access to one.

Why can the problem be solved by computational methods

Thinking abstractly

I can use the computational thinking abstractly method to simplify and reduce the information about the renter and the book loaner. Information such as what height they are or what eye colour they

may have are not required or necessary for loaning and renting a book. I can also use abstract thinking when displaying the books, as the users will only be needing the name, picture, description and the price of the book. Not when the book was made or if it is heavy or light.

Thinking concurrently

My website will be needing to use this process of thinking as it will be needing to implement parts of my solution at the same time (concurrently). An example of when my website will be needing this is when the user is viewing the dates of when a book may be rented out for. This process will need to be concurrent as the calendar which the user is looking at must continuously update while being used by the user. This is so that when the user comes to book it there will be no double bookings of it, because of a different user booking it just before them.

Thinking Logically

Thinking logically will allow me to identify decision points for branching or iteration. It can also help me discover the complexity of the algorithm needed. All the pages of my website, such as the login, sign up, home, search and checkout pages, are all decision points. This is because there are numerous options on each page for the user to select. Branching can be used on the home page as there will be a decision for the user to either view the non-fiction books or the fiction books, whatever one the user selects they will show very different results from one another. Iteration would be used for the login, sign up and checkout areas. This is because it must keep looping the page until the user has filled out all the required information.

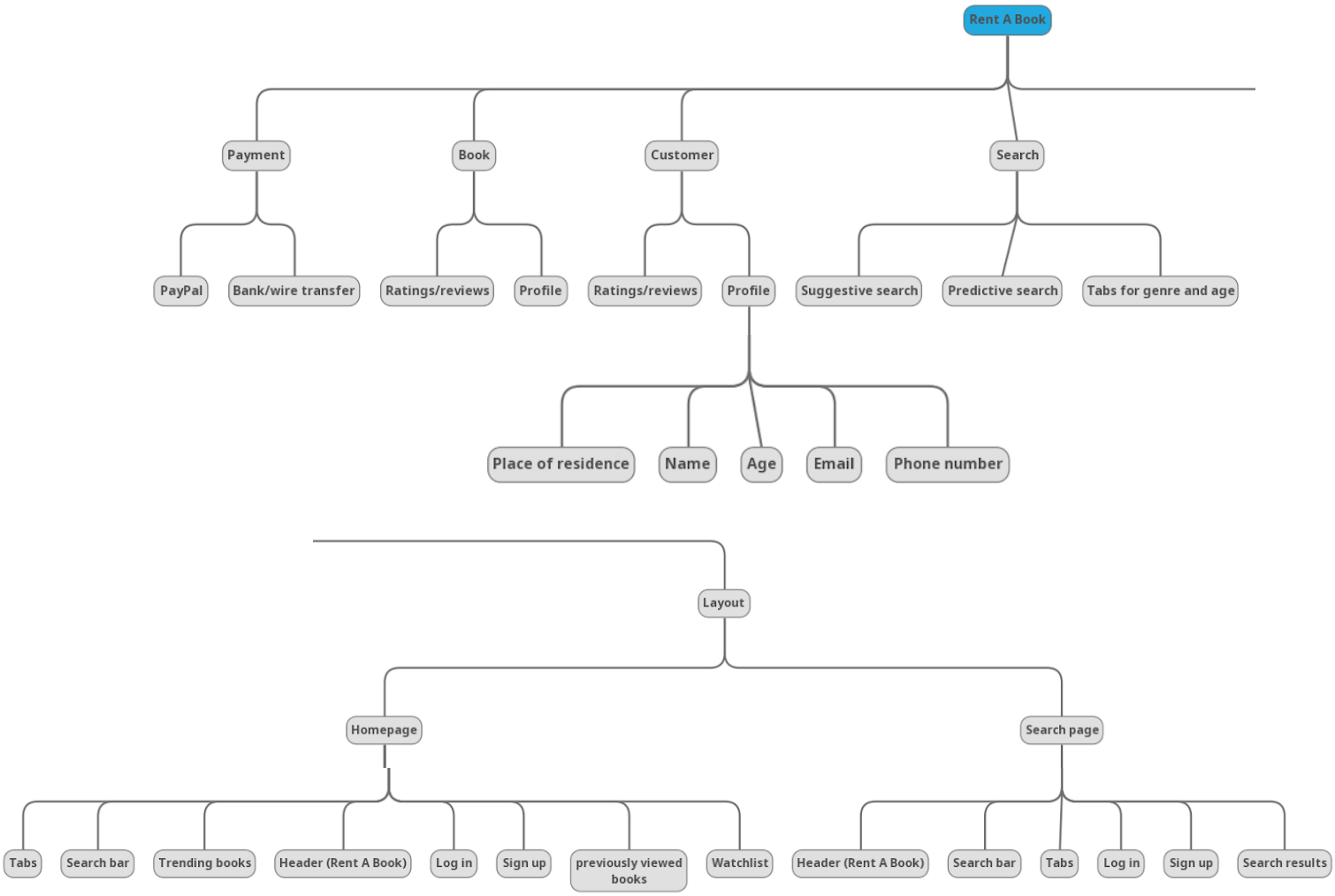
Thinking ahead

Thinking ahead is when you identify preconditions, the inputs, outputs and reusable components of a system such as a website. My website would benefit from thinking ahead as it would be used to make the user's experience of using the website more pleasant and enjoyable. I would use thinking ahead to help decide what should be displayed on each page. The search bar to help the user find a book would be shown on the homepage and all the search result pages including the product pages. It would not however be shown in the pay out page as the user is paying for the book. The tabs containing the different genres and age ratings would be on display on only the homepage and the search result pages. This is because the user may want to completely change their search parameters. Thinking ahead can also be used in the search bar to implement a predictive search to help the user find books they may like but never have thought of. I can use ahead thinking to help me think of inputs and outputs before I start coding the solution to my problem. I will be using thinking ahead by programming a feature that will suggest/predict what type of book they may want to read next.

Thinking procedurally and decomposition

Using the process of thinking procedurally and decomposition has allowed me to break down the problem into the main areas. This helps me when planning, coding and testing my solution to the problem as I can tackle them individually. There are five main parts to the problem, the layout is one of the most important parts. This is because if the user finds the layout of the website hard to use or a hassle, they are not inclined to visit the website again. To prevent this, I must have all the different features the user may want to use on each page. Thinking procedurally has allowed me to easily see what is needed on each of them and the reusable components for them. Also in thinking this way, I have figured out that my website will need a profile on all the books and the customers. This is so that when viewing a book, the customer may see what the book is about the reviews on it and the loaner, and the dates available for it. Also, customer profile would be used by the loaner for them to decide if they would like to loan the book out to them as they may not want to loan it out to

someone who has damaged books they have rented out before. Performing this thinking process on the search part of this problem has allowed me to decide on the two major search methods the first one is the tabs which will allow the user to select what genre book the want or filter it by age rating. The second method and probably most popular would be the use of the search bar as it will allow the user to search for a specific book or series. This means to help them search for the book I will need to apply predictive text to the bar. This will speed up the search process as the user will not need to type out the entire title or the author's name. Also as this will most likely be the most popular search method, I think it would be wise to implement a suggestive search to it as it will suggest books which the user may like but have never thought of.



Research

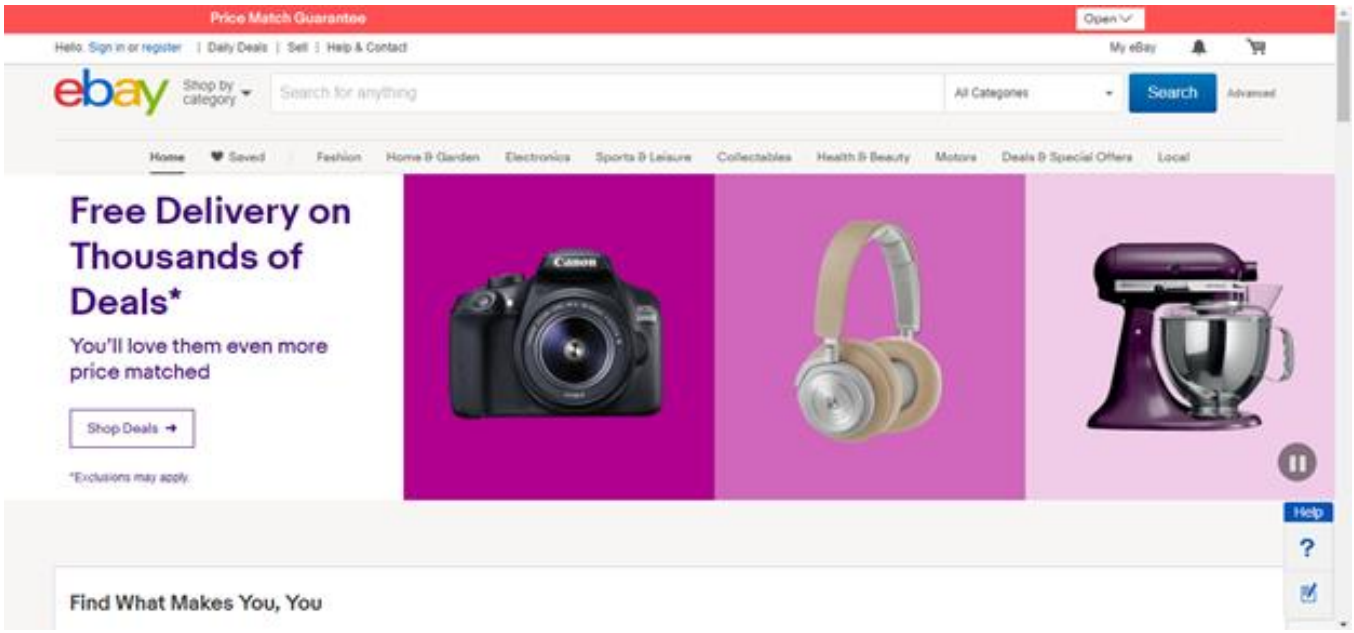
eBay

This website is the closest to what my solution will be like. This is because eBay creates a platform for people to buy and sell items to each other around the globe. My solution however will be to rent books to each other instead of buying and selling them, around the globe.

Homepage

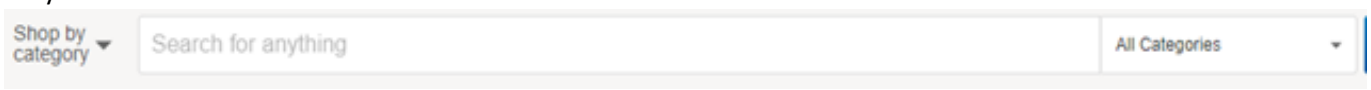
The homepage of eBay allows the user to easily navigate around it. This is because it clearly shows the user what page they are on by having the name of the page standout, it also has a clearly shown search box that the user will easily see. Increasing the user's satisfaction as they do not have a hassle

when searching for items. eBay also has a list of tabs that will allow the user to search for items specific to things such as Fashion, Electronics, Collectables etc.



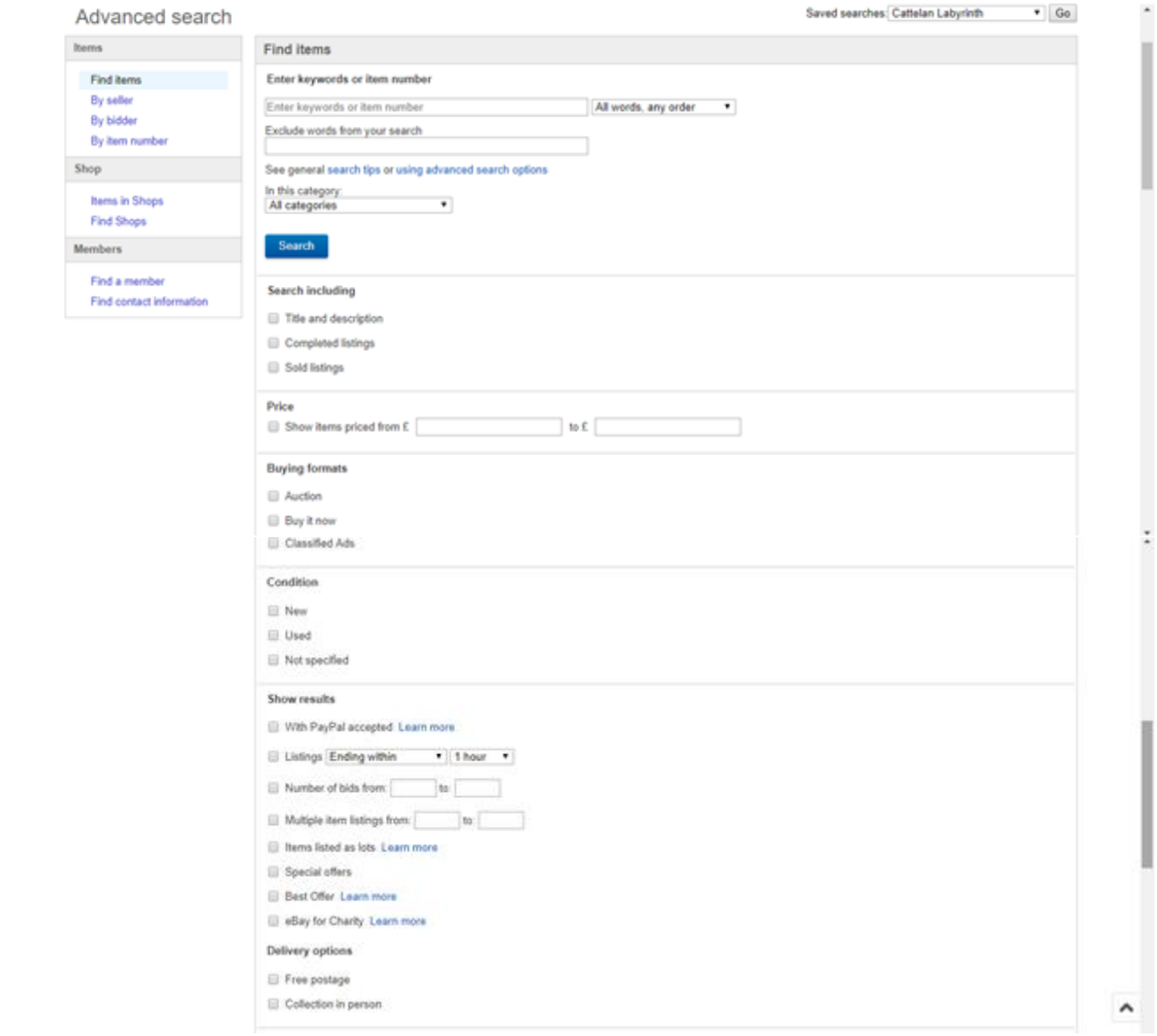
Standard search

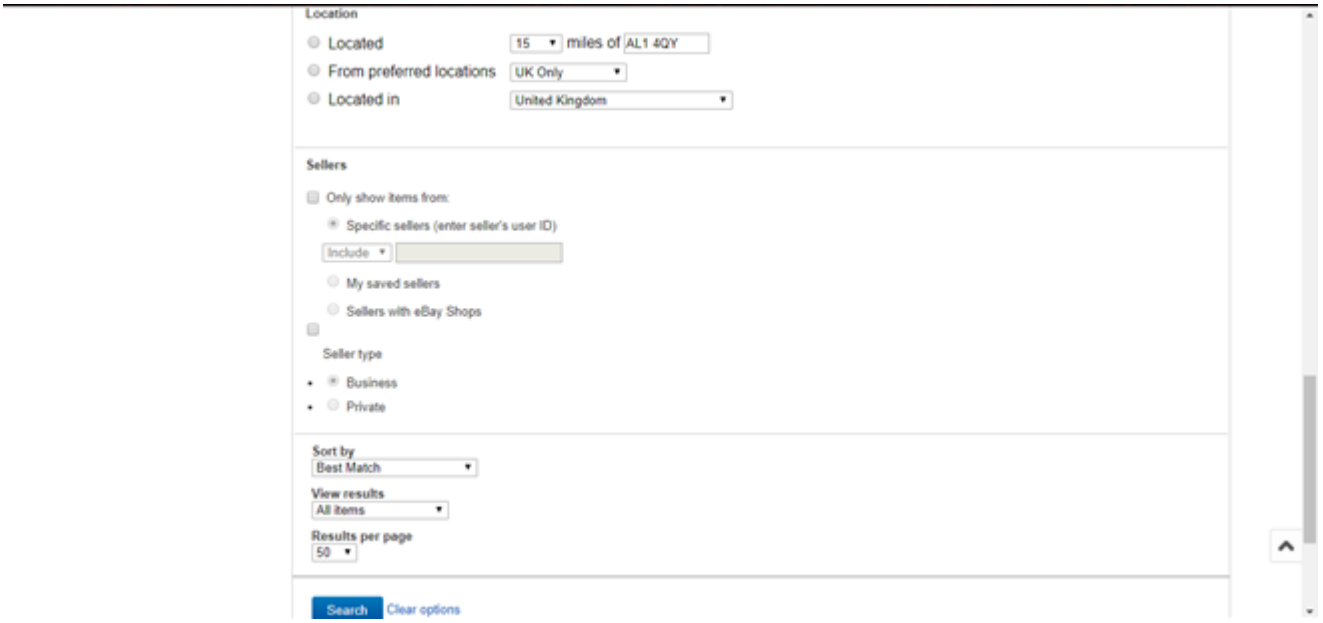
On the homepage of eBay, it has a standard search bar which gives the option to the user to choose what category they want to search in. The advantages of this is that it is quick and easy for the user to view a range of products relating to the search criteria stated. However, if the user is searching for something even more specific, they would need to go through all the search results to find what they want.



Advanced search

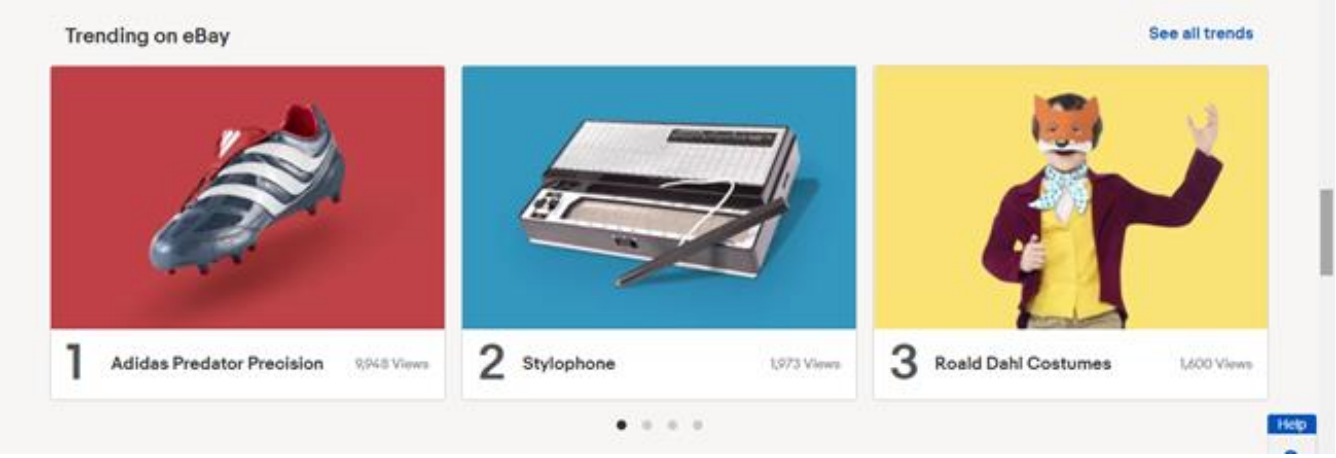
eBay does have an advanced search option which allows the user to narrow down the search parameters. The advanced search option allows the user to choose the amount of money wanted, location and even the condition of the item, etc. This option can be very useful for the user as it can help them find a specific item with the correct specification, without the need of physically going through hundreds of thousands of different choices.


Advanced search
Saved searches: Cattelan Labyrinth
Go
Items
Find items
By seller
By bidder
By item number
Shop
Items in Shops
Find Shops
Members
Find a member
Find contact information
Find items
Enter keywords or item number
Enter keywords or item number
All words, any order
Exclude words from your search
See general search tips or using advanced search options
In this category:
All categories
Search
Search including
Title and description
Completed listings
Sold listings
Price
Show items priced from £
to £
Buying formats
Auction
Buy it now
Classified Ads
Condition
New
Used
Not specified
Show results
With PayPal accepted Learn more
Listings Ending within 1 hour
Number of bids from: to:
Multiple item listings from: to:
Items listed as lots Learn more
Special offers
Best Offer Learn more
eBay for Charity Learn more
Delivery options
Free postage
Collection in person


Location
Located
15 miles of AL1 4QY
From preferred locations
UK Only
Located in
United Kingdom
Sellers
Only show items from:
Specific sellers (enter seller's user ID)
Include
My saved sellers
Sellers with eBay Shops
Seller type
Business
Private
Sort by
Best Match
View results
All items
Results per page
50
Search
Clear options

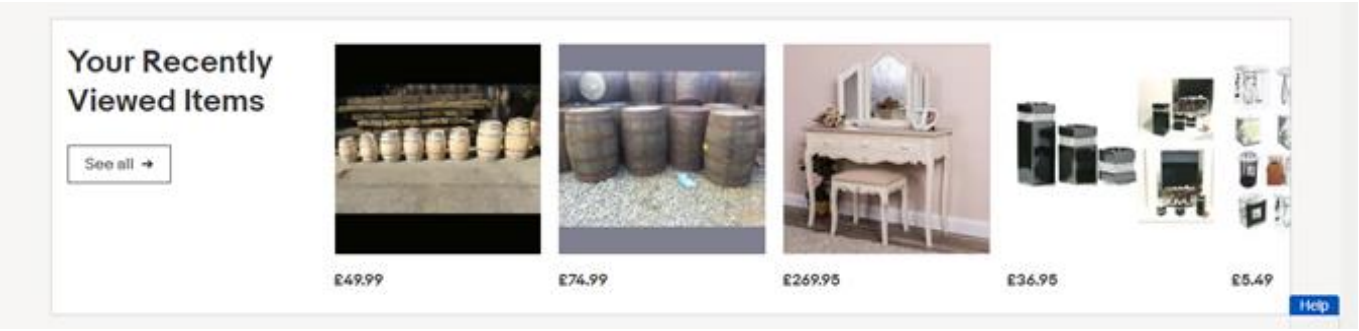
Trending

There is another feature which is very useful for websites such as eBay, that is called the “Trending on eBay” feature. This allows the user to view all the most viewed/ brought items that eBay has to offer. This feature is shown on the homepage for those who are just browsing to catch their eye on the new, trending things as they do not want to be left behind on the trend. The advantages of this is that it allows eBay to show the user things they may not have thought of.



Recently viewed

This is another useful feature for the user as it can show them what they have previously looked at/ searched for. This is useful for them because it allows them to go back to previous items and it can double as a watch list. The advantages of this is that it allows eBay another chance at promoting the item which the user has been looking at, increasing the chances of them purchasing that item, increasing eBay’s profits. This feature could tend to become more difficult than anticipated and I should not waste time trying to get it working when it is not a major feature of it.



Amazon

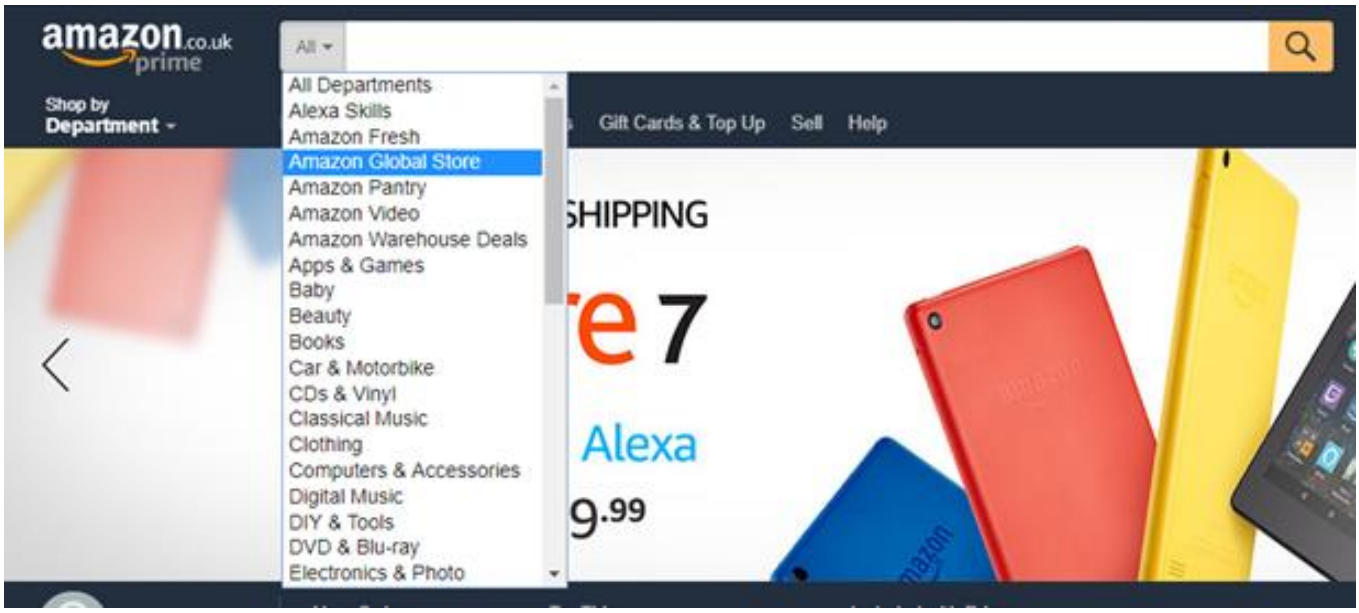
Amazon is another similar website to the one that I will be developing. However, it works with other companies and sells their items, instead my website will be loaning between people not a company. Although my website will share some features and search methods used by Amazon.

Homepage

As you can see the Amazon homepage is very similar to eBay’s homepage as the search bar is very clearly shown and the title of the page standout as well. The layout of the homepage is very clear and makes it easy for the user to navigate around and find the items they are looking for.

Search method

The amazon search bar/ method is one of the best when coming to search for an item. First, the search bar allows you to select what department the item you are looking for is in, plus if you do not know what department it is in you can select all departments to search in all of them. The department of what each item is in, is underneath the name of the item. The search bar has also got predictive search and suggestive search implemented into it. Predictive search allows the user to see popular searches based on what they have already typed in. It can help the user find what they are looking for quicker by saving on how much typing the user does, as they would not need to type in the full name. Suggestive search helps the user find what they are looking for if they have misspelled a word in the search. It does this by showing results like what you tried to search for with a spelling suggestion. It also means that if you search for something you will not be told that 'there are 0 products matching your search'.



Suggestions

Amazon has another great feature for their customers to use. This feature gives suggestions to the user based on what type of things they like viewing or have brought. The suggestions can also be

shown through categories, such as the below one which is for books. It can help suggest items they may not have considered before but has been brought by similar customers to themselves. This is a handy feature especially when stuck for a gift idea. However, replicating a suggestion area may be too time consuming, as these companies have had years to develop these features.

Survey

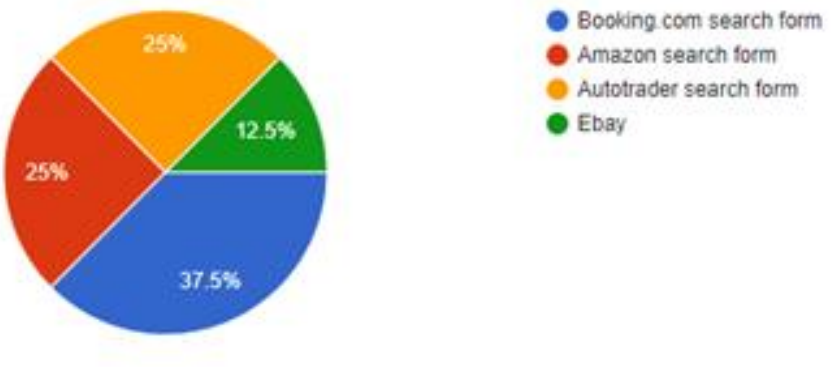
What current features do you currently like from shopping websites such as eBay and Amazon?

Many of my stakeholders have said in response to this question that they like how clear and easy it is for them to navigate around the website and that they layout is consistent throughout the website, i.e. The name and search bar at the top of the page and a list of categories down the left-hand side of the page. This tells me that I should put in a great deal of thought into the design and layout of the webpage so that the users will have a pleasant experience when using the website. Others have said that they like the idea of being able to search by category as they like browsing the webpages and seeing what they have. They also like being able to see what items are trending now. This show that I should consider adding genre choices somewhere in my website as a good amount of my stakeholders like to browse websites to see what they have.

What search methods would you like to see in a book lending website?

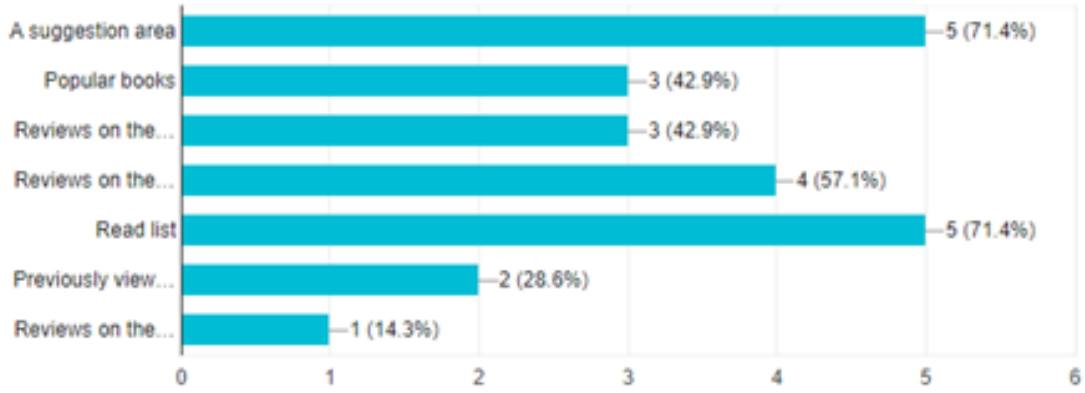
When asked this question, “What search methods would you like to see in a book lending website?”, lots of my stakeholders have said again that they would like the option to be able to search by clicking on different genres. This shows that I should add this feature into my website as it has been suggested more than once and my stakeholders like using it. Other have said that they would like to have a search bar so that they will be able to search for a specific book. This is a feature that has been also used by Amazon and eBay, telling me it is a feature that I should consider incorporating into my website.

Which search form layout do you prefer?



A majority of my stakeholders, 37.5% of them, like the clear, colourful and simple search form of the Booking.com website. While only 12.5% of my stakeholders like the colder and more formal search form of eBay. This tells me that I should make sure that when designing the search form that I keep it very user friendly and not have too many different search variables the user can change, unlike Amazon and Autotrader.

What features would you like to see on the book lending website?



A huge majority of my stakeholders have said that they would like to have a suggestion area or have a read list so that they will be able to save books that they want to read later. These are two good ideas however a suggestion area may be a bit tricky to make however I should be able to make a read list. Other features my stakeholders would like to have, was being able to write reviews of the books and their owners.

What features of existing websites do you not like?

Lots of my stakeholders have responded to this question, “What features of existing websites do you not like?”, saying that they do not like the use of popup adverts and that they would tolerate still side adverts as they would not slow down the loading of the website. I should consider this in the future as I would want to keep the website running smoothly for all the users. Other stakeholders have said that the did not like the layout or colour scheme of certain website as, they were hard/confusing to use and navigate around, or the colours they used were hard on the eyes. I must take on board these points as it tells me that the design and layout of the website is an important feature to a successful website.

Features of the proposed solution

The website I am going to create must be easy to use and have a range of different features. Features such as a suggestion area which recommends/suggests other books the customer may like because of what they have searched for or previously rented. Another feature may include allowing the user to filter a search rather than search for a specific one. They would be able to filter it by what type of genre they want, what date they would like a book for or even a specific author. I will have a predictive search bar to help the user to explore the range of different books on the website. This will help the user because it will show trending books which they might not have thought of renting before. Also, I could have an automated messaging system that can text or email the person renting the book to remind them to return the book the next day to the owner. However, I will most likely not have enough time to add this feature but if I do have time left, I will try to do it.

Feature	Justification
Search bar	A search bar should be a part of my solution. This is because it is an easy way for the users of the website to search for a specific book via its name or description. Another reason it should be a part of my solution is that a majority of my stakeholders have stated that they find it extremely useful to use as a searching tool when on websites such as Amazon and eBay. They also said that they would like to be able to use a search tool, such as a search bar to find books on my website.
Suggestive search	I will add this feature into my solution because it is the most commonly used tool by customers when searching E-Commerce websites. It will enable the user to find the book they are looking for more quickly as it is able to search for similar books if the user spelled the name wrong or if the search criteria is a bit off. However, this could become quite complex to make, as the companies such as Amazon and eBay have had years to develop theirs, so it could be just out of my scope.
Genre categories	This is a very handy feature to have on my solution as it will enable the users to browse the website by category, instead of going through everything or searching for specific books. A good few of my stakeholders have said that they like browsing websites to see what they have to offer making it a 'must have' feature in order for the website to be successful.

Trending	The trending feature is another common feature in E-Commerce websites. This feature will allow users to view what is currently the most popular books rented. This is useful, especially if a customer is just browsing the website and is not looking for anything specific yet. This feature could become time consuming, although I think I would have the entire homepage showing trending books instead of a small section which is what eBay and Amazon have.
Read list	This feature would allow the user to keep track of books they wish to read in the future when it becomes available. Lots of websites have features similar to this, like eBay which has a watchlist that is used many times by users, making it a useful feature to have on my website.
Suggested	Suggested books feature will help the user find a book which they may not have considered before or have not heard of. It is a useful feature as it can increase the chances of a customer renting out a book, instead of leaving the website empty. Again, this feature can quickly become complicated to build so it is most likely out of my scope for this project.
Basket	This feature must be a part of my solution as it is required, so that the user will be able to rent out the books they have selected.
Predictive search	The predictive search feature is also very common among E-Commerce websites. It is a very useful tool for users, especially if they are using a device which does not have a keyboard for them to type on. The feature will be able to show the user popular searches based on what they have already typed in. It can help the user find what they are looking for quicker by saving on how much typing the user does, as they would not need to type in the full name.
Sign up area	This is another 'must have' feature as it will allow the website to get the necessary details so that the user will be able to upload books onto the

	website, and so that the website will have the correct contact details to inform them.

Software and hardware requirements

Software

To develop this website, I will be needing to use programming languages such as HTML, JavaScript and CSS. These three languages will allow me to build an interactive and professional looking website, which the public can use to rent and lend out books. I will also be needing to use the database management language called SQL. I will be using this to search, sort and all together handle the databases which will hold the user's information and information about the books. I will also require a server for the website to be held on and to store the user profiles and books for rent. To build the search boxes for the website I have an option between two coding languages, Perl and PHP. There are many advantages and disadvantages for both, PHP for example has a lot more security holes popping up each week compared to Perl, however PHP is faster. I'll be using ASP.NET that is server-side technology for creating dynamic web pages. There are many important features ASP.NET such as the server-side code of the pages are fully compiled and executed, instead of being interpreted line by line. I'll be using visual basic.net as it is one of the most popular programming language for developing presentation-tier logic for websites developed with the .NET framework. My stakeholders will require an internet connection and either google chrome or Microsoft Edge to be able to connect up to the website.

Hardware

Requirement	Justification
Monitor	A monitor is required both for the development and deployment of the website as it is a way for me and the users to view the website.
Mouse	A mouse is also needed as it is used by the user to select what books they may want to rent out. It is also used to help them navigate around the website and helps me with the development.
Keyboard	The keyboard is needed to allow the user to input text into the search area and when logging in and signing up. It is also a major requirement for the development of the website as it allows me to type in code for the website.
7GB of RAM (Combined web and database server)	This is the amount of RAM needed for the server that will hold my website and its databases.

Processor 4 x 1.6GHz CPU	This is the recommend amount of processing power needed for the combined web and database server.

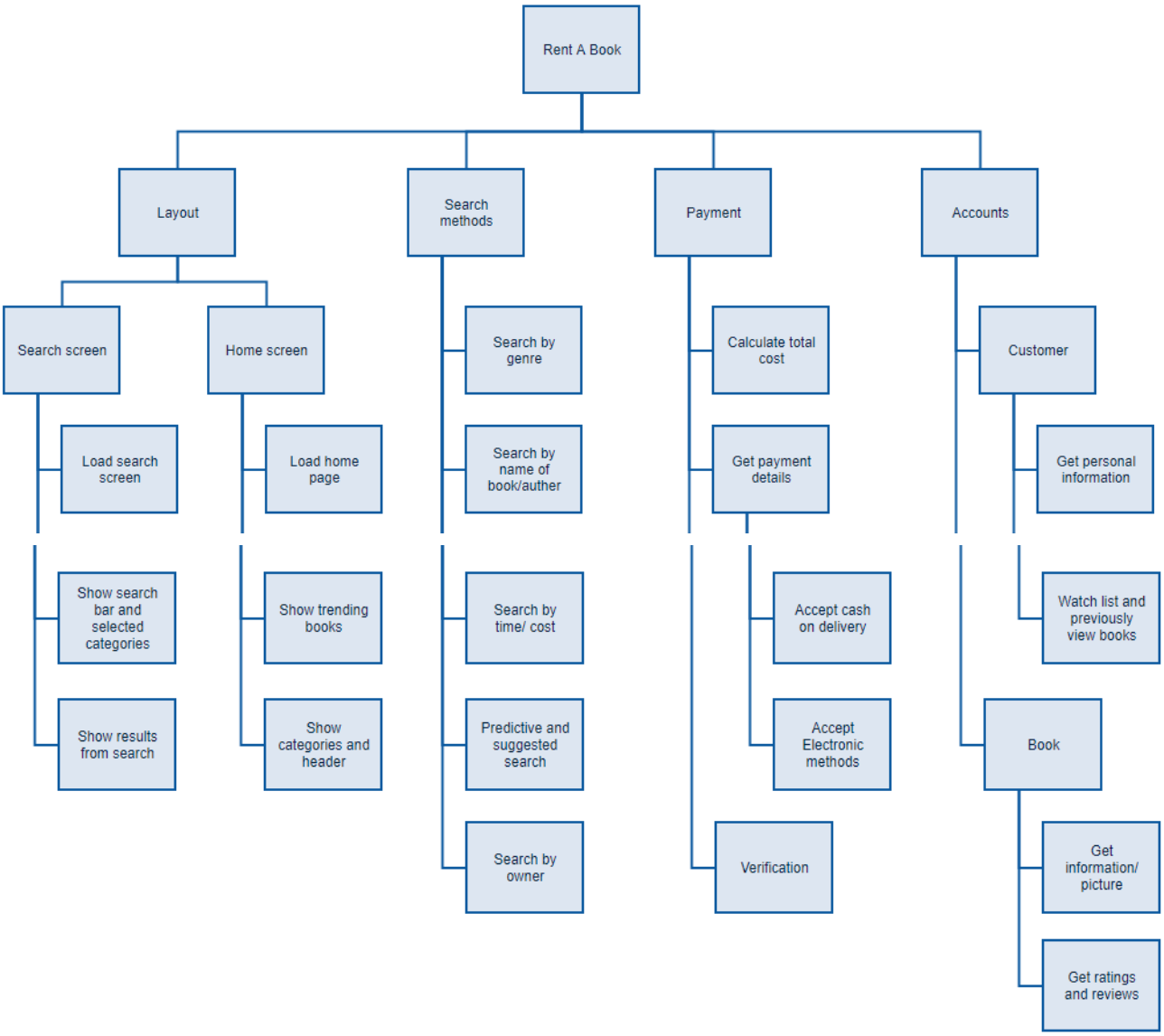
Success criteria

Requirement	Justification
My website must allow the user to somehow purchase a book.	This is the main feature my website is to do, if it is unable to do this, then my solution to the problem does not work.
Payment must be able to be done through PayPal	PayPal is the most commonly used software for people to use when buying online things, which is why it is a handy feature for the customer to have and making it necessary that my solution can integrate with it.
Sort by genre	This method of searching through the database has been mentioned in my survey by the target market. It is also a common feature in many E-Commerce sites including the two I have investigated, eBay and Amazon. Fiction and non-fiction are the two types of books, the list of genres should appear underneath the type selected for the user to choose
Predictive search	Predictive search is also very common among E-Commerce websites. I came across this when investigating Amazon, and it is a very useful tool for users, especially if they are using a device which does not have a keyboard for them to type on. Predictive search will help the customer find books by showing them titles of books similar to what is being typed in
Suggestive search	I discovered this alongside predictive search when analysing Amazon. Suggestive search is probably the most useful tool for searching, which is why one of my target market mentioned this in the survey. This is because it helps you find what you are looking for if your spelling is wrong or if your search criteria is a bit off. This feature is handy as it also enables the user to view books they may like but never thought of before.

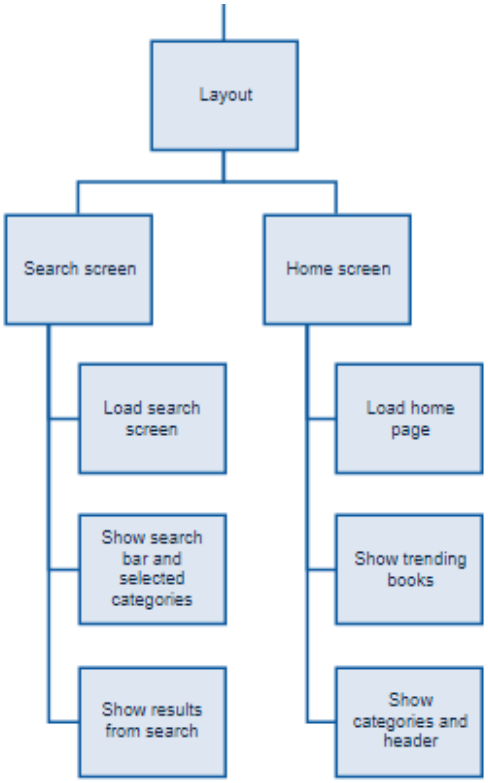
The user must be able to sign up	This is another important feature the website must be able to do as it enables the customer to sign up and rent out books
Sign up and login area should be able to handle errors	The signup and login areas should be able to handle errors because if the user types in an invalid email address or a different piece of information, the website should be able to realise this and let the user know of the error. It should also be able to check that every piece of required information has been filled in correctly.
The sign-up area must be able to check and save to the correct database	The sign-up area must be able to only access the customer database. This is so that when the user clicks the submit button their information will not be saved in the book database instead. It must also be able to search through the customer database to make sure that there are no existing databases too like the one the user is trying to make.
The login area must be able to open the correct account	The login area must be able to open the customer database, check if the account exists, and then open for the user to use. This is so that no fake user may use the services of the website or pretend to be a different user. It must be able to open their account so that they may be able to see suggested books and be able to update details such as their Email addresses.
Read list	In the survey, people liked the idea of being able to save different books they wish to rent out for a later time. This is a useful feature as it enables the user to keep track on what they want to read without the worry of forgetting one of them or trying to write them down.

Design

Overview of the system (systems diagram)



I have broken down the problem into its smaller process/ functions and presented it through a systems diagram. By doing this it has allowed me to visualise what I must do for my solution so that it will be effective against the problem. The main features are the layout of the main pages, the search and payment methods and the accounts. I then broke down these four features further to reveal more small and manageable processes for them.

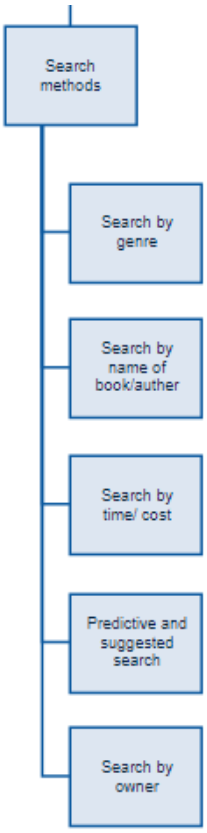


Layout

I have broken down this feature into the two main pages, the search page and the home page. The search page will be where the user will see the results of the queried/ searched book. Therefore, one of the process I have broken it down to is to show results from search. This process would be reading the returned data from the query and present it in a user-friendly way. The page will also need to show what the specifically searched for, in case they accidently pressed the wrong genre and was not to sure if they clicked the correct one.

The home screen will be the very first page the user will see. This means it is the most important page in the entire website as it must catch the user's interest, because of this I had to think of what the customers would expect to see on it and where these features should be found by thinking ahead. To do this I needed to think logically and use the data I acquired during the analysis stage of this project to determine what processes will be needed for the home page.

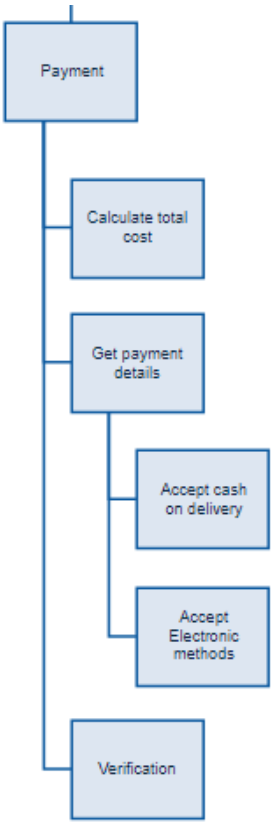
One of the processes for the home page will be for it to show the currently trending books as it was a feature liked by most of my participants for the survey I performed in the analysis section. By having this on the home page it will allow the user to see what the most popular books are for rent and maybe even impress them with the latest books on rent for low prices. The home screen will also need to show the different categories for the user to choose and the header Rent a Book to advertise the website to everyone looking at the page. The list of categories is an absolute must for the home page as the research I did in the analysis section showed me that all the successful websites have some sort of category list for the user to browse through and every single participant from my survey wanted this feature.



Search methods

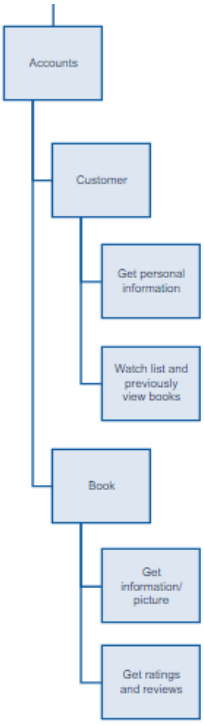
This feature has been broken down into five processes. The first process is the search by genre, this search method will allow the user to browse the website by querying the database table through different genres such as action, romance or cooking. The next search method would be search by name of book/ author. This process would mean the program must search for a specific author or book, however if it is not found it must then suggest similar names of authors and books, in case the user misspelt it. Another process for the search methods would be for the user to search by time or cost. This would allow the user to find books for a specific time frame and if they only want to spend a certain amount of money for it. By thinking logically I was able to deduce An additional process for search methods. The additional search method could be to by owner. This would mean that the user can search for a book owned by the same owner or in the same location. Meaning the user will not be renting a book which is three thousand miles away when another exact

same book maybe 5 miles away. The predictive and suggested search process would help the user browse the database a lot quicker and efficiently. By this I mean that it will help reduce the chance of a spelling mistakes and allows them the chance to view books which they may not have considered.



Payment

For the payment feature there are three processes and one of them I broke down further. Calculating the total cost is the first process as it allows the customer to see how much they are about to spend on renting the books. If this process is not done correctly the website may over or under charge the customer resulting in many issues later. The second process is to get the payment details from the user. This process was broken down further as there are two ways of payment transactions, cash on delivery and electronic methods. Electronic methods would mean using a bank transfer, direct debit or even PayPal. After all this the verification process begins. The verification process will check to see if all necessary payment details have been filled out correctly and that there is enough money in the account for the transaction. If this is not performed adequately then the transaction may not go through or the money being transferred might go to the wrong account by mistake.



Accounts

There will be two types of accounts, a customer account and a book account. The customer account will contain personal information such as name, address and contact info to alert them when someone wants to rent out one of their books. The account will also contain the customers watchlist and what they have previously viewed. This will allow them to keep track of what they have read and keep an eye on a book for a specific date.

The book account will hold information such as the name of the book and its author. It will also hold the name of the owner and there will be a photo of the book, the condition and language will be part of the information as well, along with other things. The account would also contain the book's ratings out of five and reviews of the owner and book. This is so that potential lenders will be able to see previous user's opinions on it before renting it out themselves. The dates of availability will be in this account as well.

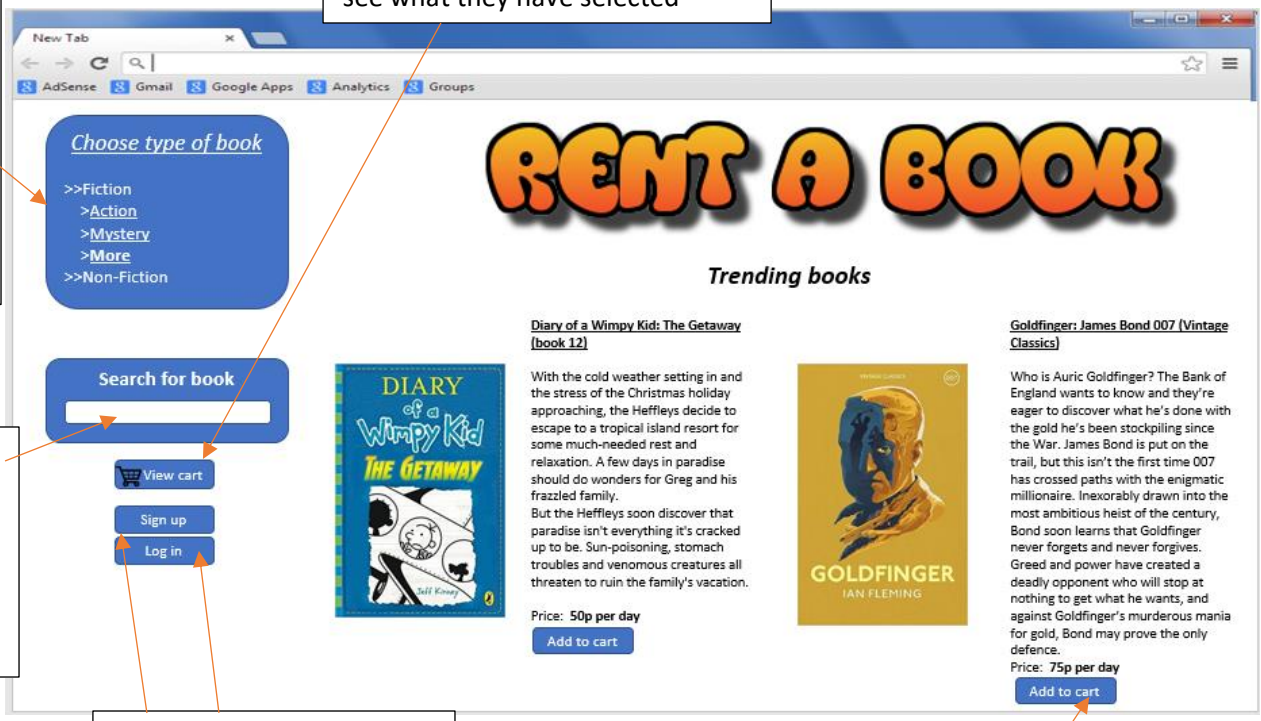
I applied thinking abstractly to break it down to these manageable and controllable steps. This is because it allowed me to focus on what the main features each account need, I was then able to simplify it down even further by allowing the two types of accounts to share them, reducing the amount and simplify the coding even further.

Proposed Screen Designs and usability features

Homepage

This would be the list of genres that they user can choose from to find a type of book they want.

The view cart button would allow the user to open the basket and see what they have selected



The search bar will enable the user to find a specific book

The Sign up and Log in buttons will allow the user to easily create an account and be able to open their account

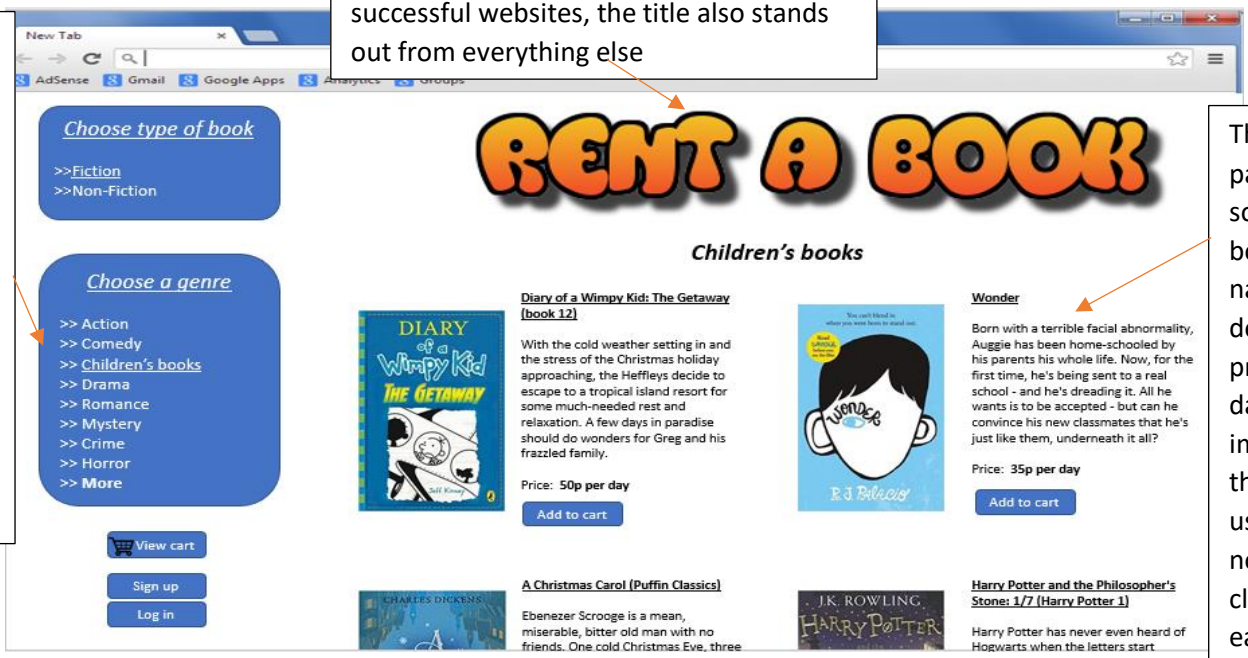
This button will allow the user to add the selected book to their basket

As you can see, my proposed homepage design has taken on board what my stakeholders have said in the survey. I have designed my homepage so that it will be easy to learn as it follows common design features that Amazon and eBay have that cross over. Such as how the genre categories are along the left-hand side of the page and that the title of the page is standing out and at the top of the page. The colour scheme of the homepage and the use of big buttons will make the page more engaging. Also, due to its simple design, the page would become effective as the users will find it easier to navigate around it, which will also make it efficient as the user will be able to quickly get around the homepage and select a book.

Search page

The title of the website will be at the top of the page as this is the 'norm' of most of successful websites, the title also stands out from everything else

After the user has selected a type of book, another box will appear allowing them to choose a genre from that type of book

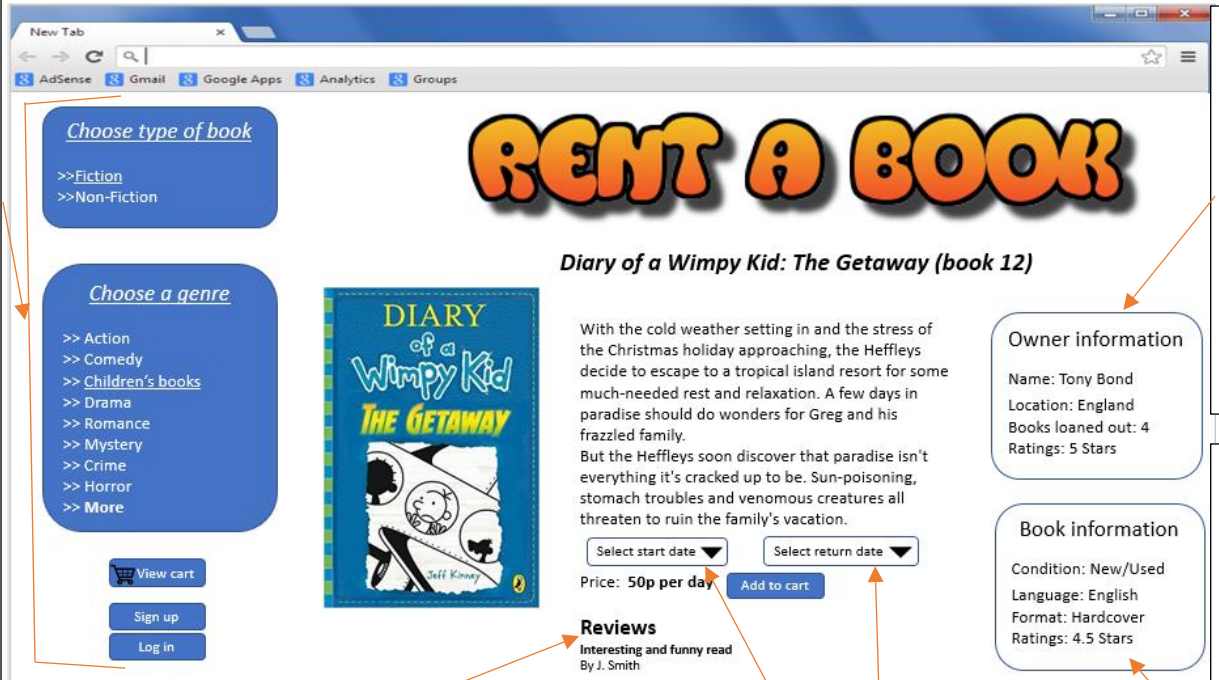


The search page will sow each book's name, description, price per day and image so that the user will not need to click on each book to find out what it is about

The search page has a similar layout to the homepage. This is an advantage to the user as it makes it easier to learn because each page follows the same set of standards, which I found out from the analysis section that my stakeholders prefer this in websites. Because the layout of it is like the homepage, it also increases efficiency as the users will know where everything is on the website as it is the same on each type of page in the website. The search page will also show the user brief details about the books shown making the website effective as the user will not need to click on each book to get an idea of what it is about.

Profile page

The choose type of book and choose a genre boxes will remain open of the left-hand side of the page so that the user can quickly and easily start looking at other books in a different category



The profile page will contain information on the owner such as their name and their ratings out of 5

The page will also have more specific details about the book such as what condition it is in

The profile page will have a reviews section that will contain the opinions upon the book and its owner from previous people who have rented it out

These are drop down calendars that will allow the user to select when they want to rent out the book and for how long

The profile page will allow the user to get more details on the book, such as the condition it is in and what format it is in. The profile page will also tell the user information about the owner, like what ratings they have been given from previous people who have rented from them. The layout of the page also follows the set standards, such as list of genres down the left-hand side, making it easier for the user to learn how to use the page. It also makes it more efficient as the user will know where everything is, meaning they do not need to put in a great deal of thought to navigate around the website. The use of the drop-down calendars will make the page more error tolerant as it will not allow the user the chance to accidently put in an invalid date.

Log in page

The log in page will only contain a box for the user to input their username into and another box to input their password into

Log in

Username:

Password:

LOG IN

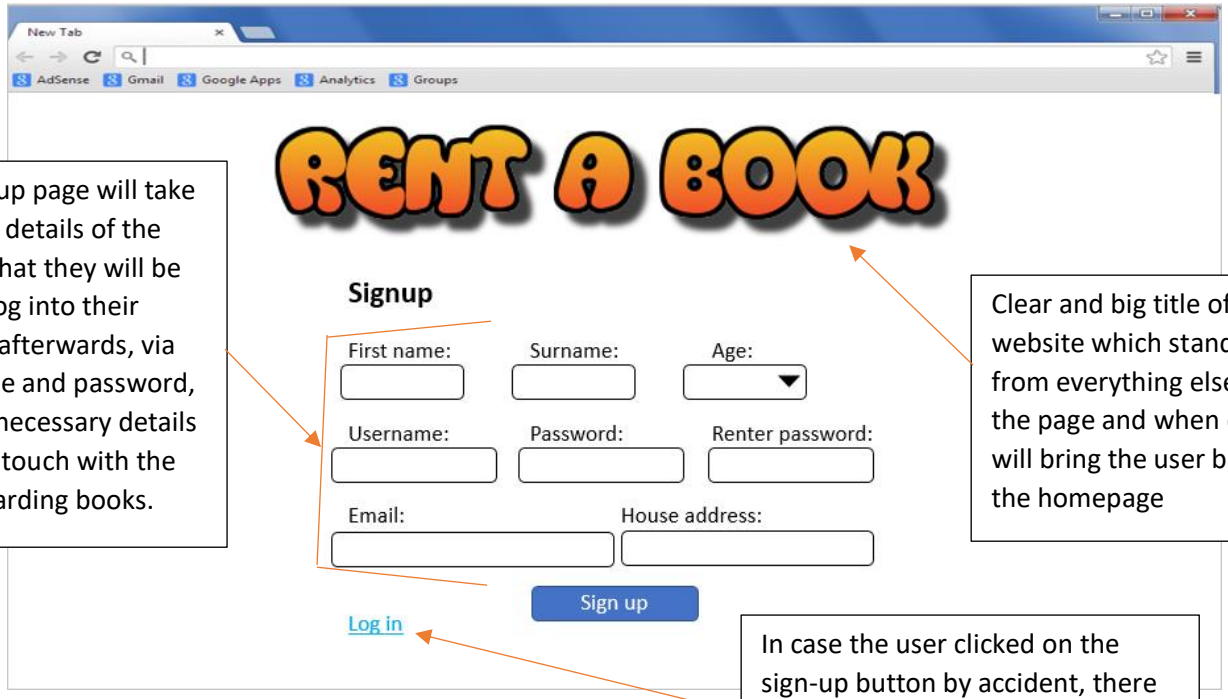
Signup

Clear and big title of the website which stands out from everything else on the page and when clicked will bring the user back to the homepage

In case the user clicked on the log in button by accident, there will be a hyperlink at the bottom of the page to bring them to the sign-up page

When the log in button is clicked this page will appear to allow the user to sign into their account using their username and password. To make the page easy to learn and engaging, I based its design on the Google page, making it clear and simple and having the text and textboxes for the user to type into, big and bold, reducing strain on the eyes. For error tolerance I will make a pop up message explaining to the user that they have either typed in the wrong username or password, and if they wish to try again or sign up. There is also a sign-up hyperlink at the bottom of the log in page incase the user accidently clicked on the login button. The page is also efficient as there are no unnecessary details that also need to be filled in, so that the user can sign in.

Sign up page



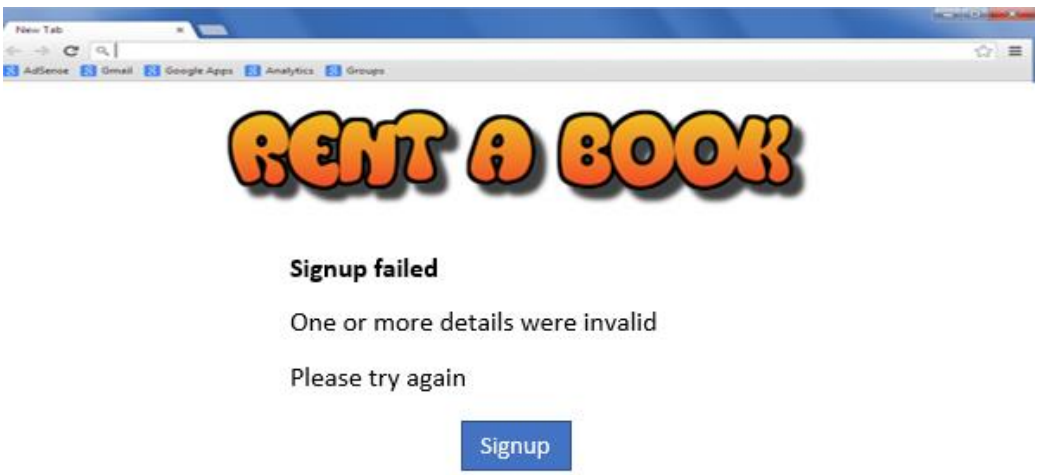
The signup page will take in all the details of the user so that they will be able to log into their account afterwards, via username and password, and the necessary details to get in touch with the user regarding books.

Clear and big title of the website which stands out from everything else on the page and when clicked will bring the user back to the homepage

In case the user clicked on the sign-up button by accident, there will be a hyperlink at the bottom of the page to bring them to the log in page if they already have an account

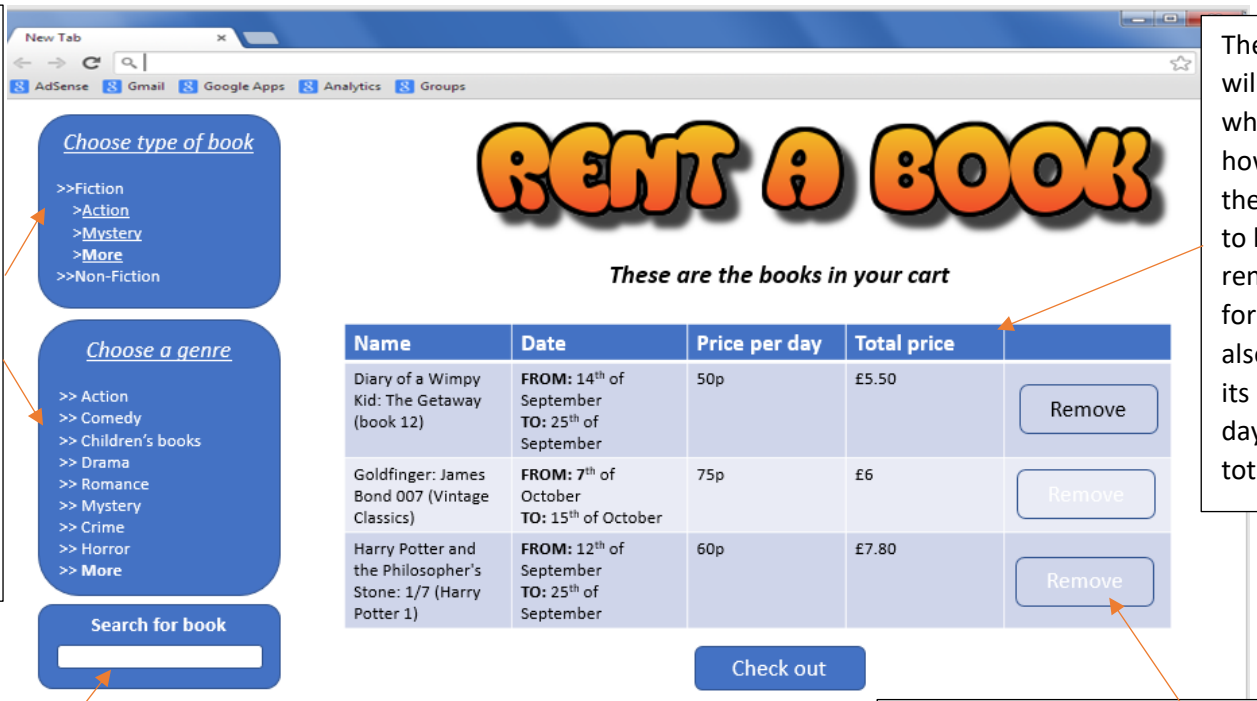
This page will allow the user to easily create an account due to it simple design making it easy to learn and efficient as it is not complicated to use. I will also incorporate multiple error tolerance features into it, such as the drop-down box for the user to select their age and a renter password box so that the user does not accidently hit a button when typing in their first password. The form is also effective as it easily gets all the detail the website needs to get in touch with them regarding the books.

Popup error box



Check out page

The two boxes allowing the user to choose a type of book and a genre will be remain on the left-hand side of the page as like the other pages.



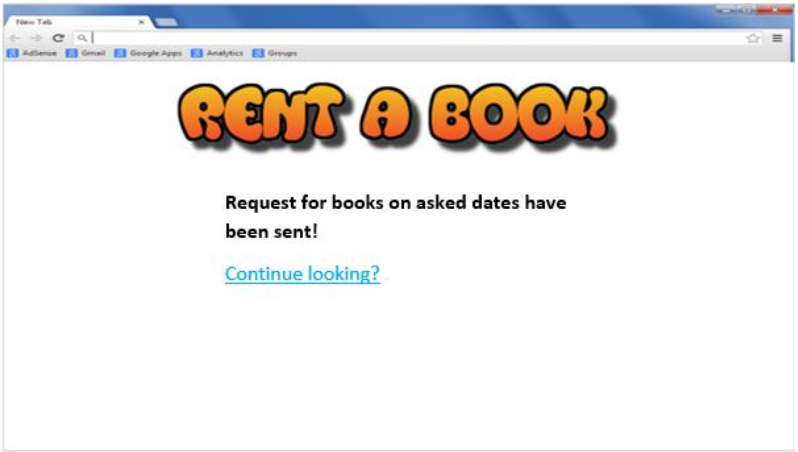
The basket will show when and how long the book is to be rented out for. It will also show its price per day and total price.

There will also be a search bar for the user to use to find a specifc book

There will be a remove button to allow the user to take books out of their basket, in case they change their mind about it or if they added it to the basket accidently

This page will allow the user to view what they have in their basket already. I will add a remove button so that the user will be able to delete books from their basket, making the page error tolerant as the user is able to fix their error of adding the book to their basket. I have also designed it to be efficient as all the user needs to is click on the checkout button at the bottom of the basket, which is where most checkout buttons are making it also easy to learn, to rent the books.

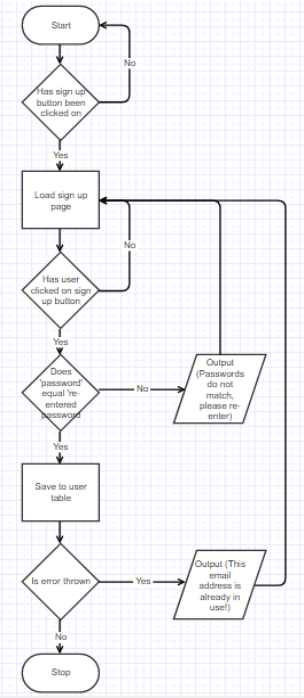
When checkout button is clicked



Algorithms

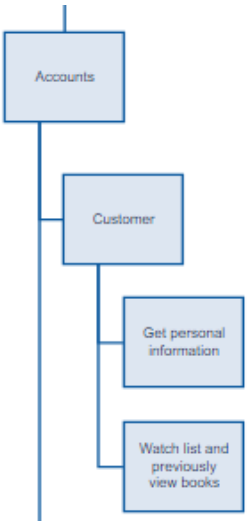
Signup

Flowchart



Pseudo code

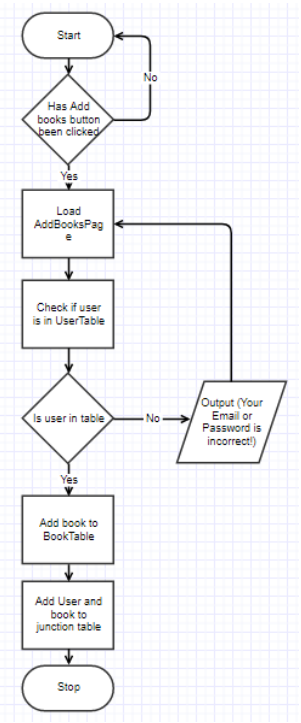
```
Procedure AddToUserTable (name, email, password)
        Insert into UserTable (
        Name = name
        Email = email
        Password = password)
END Procedure

If SignUpButton = True
        LOAD SignUpPage
        If SignUpClicked = True
                If password != rePassword
                        print (Passwords do not match, please re-enter)
                Else:
AddToUserTable (name, email, password)
```



This algorithm will allow the user to sign up an account with the website. It will first check to see if the sign-up button has been clicked, if it has it will change the Boolean variable SignUpButton to true. After that it will load the sign-up page and allow the user to fill in their details. It will then check if the user has clicked on the submit button, if so it will compare the first and second password to make sure that they are the same. If they are not it will inform the user that the passwords they have entered do not match, otherwise it will call the procedure AddToUserTable and pass in the variables name, email and password. They are then saved to the user table. I designed them this way as it will allow faster processing time and the use of iteration has led to a verification process. This means that the user can only advance if they meet the iterations requirements.

Add Book

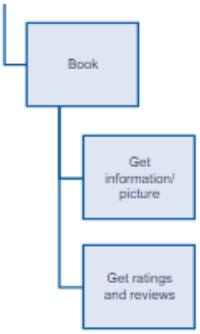
Flowchart



Pseudo code

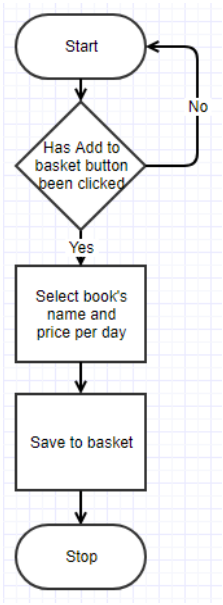
```
Procedure CheckUser (email, password)
        Where email = Email and password = Password IN UserTable
        Authentic =+ 1
END Procedure

Procedure AddToTheBooksTable (name, description, pricePerDay)
        Insert into BooksTable (
        name = Name
        description = Description
        pricePerDay = PricePerDay)
        Where name = Name
        Return BookID
END Procedure

Procedure AddToUserJunction (userID, bookID)
        Insert into UserJunction (
        userId = UserID
bookID = BookID)
                END Procedure

    If AddBooks button = True
                LOAD AddBooksPage
                If Add button = True
                        Authentic = CheckUser (email, password)
                        If Authentic != 1
                                print (Your Email or Password is incorrect!)
                        Else:
                                bookID = AddToTheBooksTable (name, description,
pricePerDay)
                AddToUserJunction (userID, bookID)
```

This algorithm will allow the user to add their books onto the website which is one of the main features of this solution. The algorithm will first check if the add book button has been clicked on by changing the variable to true, if so. Next it will load the AddBooksPage to allow the user to enter their email and password, and the details of the book. The algorithm will then check to see if the user has clicked on the submit button. After that it will then call the checkUser procedure which will check to see if the user has entered a valid email and password. If they haven't it will tell them that they have either enter their email or password wrong. Otherwise it will call the AddToTheBooksTable procedure which will add the books details to the table and return its book id. Tis is so that it can be used by the next procedure which will create a relationship between the book and the user. I designed it this way to make the process easier to code and develop as each function can be maintained and reused during the iteration that allows verification to happen.



Add to basket

Flowchart *Pseudo code*

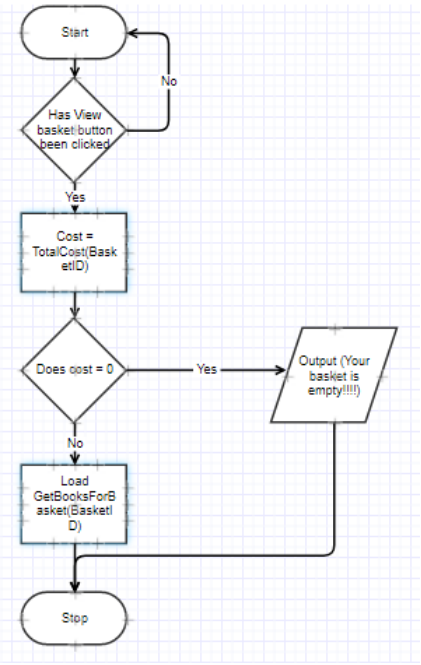



```
Procedure AddToBasket (basketID, bookID)
        Insert into Basket (
        basketID = BasketID
        bookID = BookID
        Days = 1
END Procedure


If AddToBasket button = True
        AddToBasket (basketID, bookID)
```

The add to basket algorithm will allow the user to add numerous books to their personal basket. This is important when they wish to rent them because if it does not work they will not be able to rent them. This algorithm will first check to see if the user has clicked on the add to basket button. If so it will change the variable from false to true, resulting in the calling of the AddToBasket procedure. This procedure will take in two variables, basketID and bookID, which are used to insert the book into the user's basket.

View basket

Flowchart



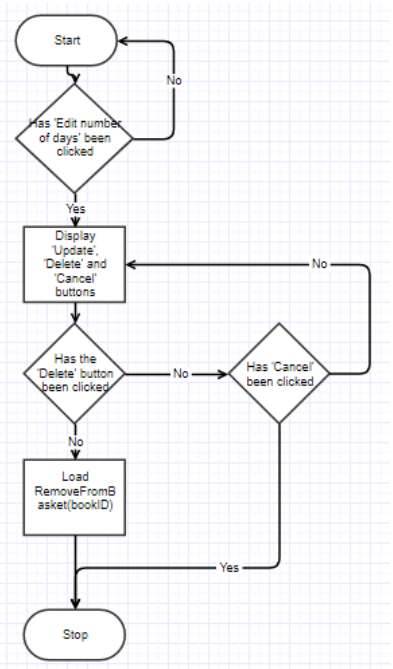
Pseudo code

```
Procedure TotalCost(BasketID)
    While books in basket != 0
        Num = book.PricePerDay * days
        TotalCost =+ Num
        Remove book
    Return TotalCost
END Procedure

Procedure GetBooksForBasket(BasketID)
    Select all bookID, Name, PricePerDay, Days, Cost
    From Basket
    Return bookID, Name, PricePerDay, Days, Cost
END Procedure

If ViewBasket button = True
    Cost = TotalCost(basketID)
    If cost = 0
        print (Your basket is empty!!!!)
            Else:
                Basket = GetBooksForBasket(BasketID)
                Print (Basket)
```

The view basket algorithm will enable the user to see what they have selected to rent, it will also work out the total cost of the basket if the user were to pay for everything in it. The algorithm starts of by checking to see if the user has clicked on the view cart button. If so, it will set the Boolean variable to true to represent this. It will then work out the total cost of the basket, using the procedure TotalCost, and return the results into the variable Cost. If the cost of the basket equals 0, then it will inform the user that their basket is empty. Otherwise it will run the procedure GetBooksForBasket, which will return all the books in their basket. After that, the user's basket is displayed to them.

Delete, cancel and update from basket

Flowchart



Pseudo code

```
Procedure RemoveFromBasket (basketId, bookId)
        Delete from Basket
        Where BasketID = basketId AND BookID = bookId
End procedure

Procedure UpdateBasket (basketId, bookId, days)
        Update Basket
        Set Days = days
        Where basketId = BasketID AND bookId = BookID

If Edit number of days button = True
        Display days textbox, Update, Delete and cancel buttons
    If Delete = True
        RemoveFromBasket(bookId)
    Else if Update = True
        If days <= 0
            print (You can only enter valid number of days)
        Else:
                                UpdateBasket (bookId, days)
        Hide days textbox Update, Delete and cancel buttons
```

This algorithm enables the user to either delete or update the basket. It will also have a cancel button in case they do not want to save the changes made to the basket. The algorithm will first check to see if the button Edit number of days have been clicked. If so, it will then display the three buttons and textbox for the user to enter the number of days wanted. It will then check which of the three buttons have been clicked, if it is the delete button, the algorithm will call the RemoveFromBasket procedure. This procedure will delete the selected book from the basket. If the update button was clicked instead, then the algorithm will next check how many days have been entered. If the days equal or less than 0 then it will inform the user that the number of days entered is invalid. Otherwise it will call the UpdateBasket procedure instead. This procedure will then update the basket to the amount of days requested to the selected book. The algorithm will then finally hide the textbox and the three buttons. This has been designed this way so that there will be less to code as I will be able to repeatedly use the functions that make up the process. This will also make it

easier to maintain as I will not need to change something multiple times indifferent places in the code, but just one area.

Variables, Data Structures, Validation

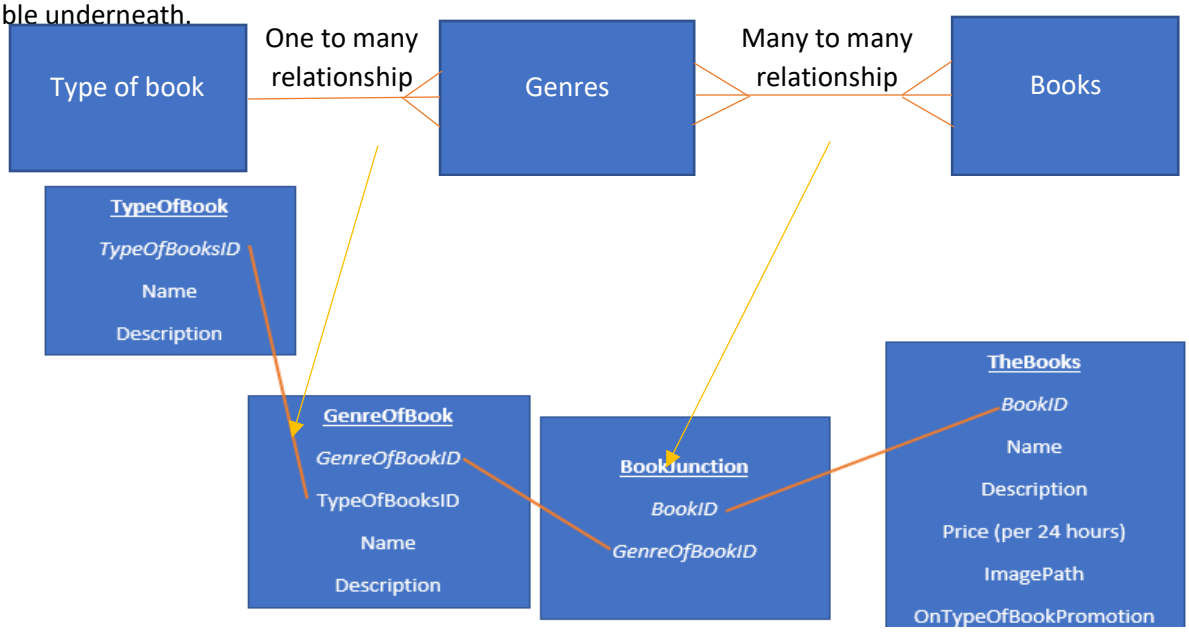
Name	Type	Justification	Validation
Catalog	Class	I created the catalog class to encapsulate the data and the methods relating to the books for rent.	N/A
GenreOfBookID	Integer	This variable will hold the unique ID for the selected genre. By doing this it will save memory space as it will not be needing to save a long string.	N/A
TypeOfBooksID	Integer	This variable will hold the unique ID for the selected genre. By doing this it will save memory space as it will not be needing to save a long string.	N/A
SearchString	String	The search bar will allow the user to type/search for any word or words, meaning the variable holding it should be a string.	N/A
Account	Class	I created the account class to encapsulate the data and the methods relating to the user's accounts	N/A

ViewBasket	Boolean	When the user clicks on the View Basket button, this variable will change from a 0 to a one to show that the button has been selected.	N/A
SignUp	Boolean	This variable must be Boolean as the user can only either have clicked on the button or haven't clicked on it.	N/A
AddBooksPage	Boolean	When the user clicks on the AddBooksPage button, this variable will change from a 0 to a one to show that the button has been selected.	N/A
Cost	Float	This variable will hold the cost of the entire user's basket.	N/A
Email	String	This will hold the user's email address, which can have a range of different characters, meaning the variable must be a string.	The string must contain an @ and at the end of it, have .com or equivalent. Otherwise it would not be a valid email address and the program will not be able to get in contact with the user. The email must also be unique to the other emails in the database.

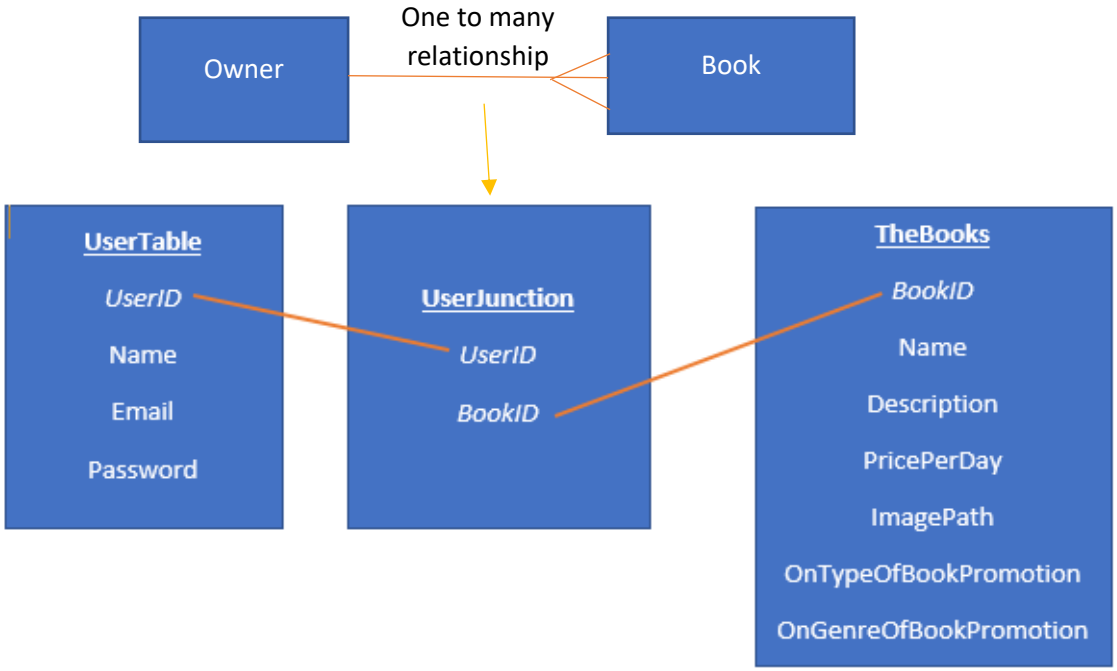
Password	String	This will hold the user's password, which can have a range of different characters, meaning the variable must be a string.	This variable will hold the users password and will be compared to the password stored in the database to check if the user has entered the valid password
Name	String	The user's name will be entered a textbox where it will be saved as a string variable, because it can contain a range of characters.	N/A
Basket	Class	I created the basket class to encapsulate the data and the methods relating to the user's individual baskets	N/A

Entity-relationship diagrams

This entity- relationship diagram represents the relationship that the books have with the genres, and the relationship the genres have with the type of books. The relationship between the types of books and the genre of books is one to many, as there can be multiple genres belonging to one type of book. While the relationship between genres and books are many to many. This is because multiple genres can belong to one book and multiple books can belong to one genre. Because it is a many to many relationships is used, a junction table is needed, which is why there is a book junction table underneath.



Another entity relationship diagram is between the users/owners and the books. This is so that the database will know who owns what book. The relationship between the users/owners and the books will be a one to many relationship. This is because one user/owner can own multiple books on the website, while that one book can only have one owner.



Iterative Development Test Data

These will be the tests I will be preforming throughout the development of my solution. I aim to meet each of the aspects being tested before moving onto the next iterative. I will only perform the tests from this table that are relevant to the iteration/ part of my website that I am programming.

Test No.	Aspect being tested	Justification	Test Data	Type of test (valid, invalid, boundary)	Expected result	Actual result	Evidence (screenshot, screencast etc)	Actions needed (e.g. bug-fix and retest, ignore etc)
1	Does it accept a valid email	A valid email address is required for the user to be contacted	test@gmail.com	Valid	Sign up/login successful			
2	Does an error appear when an invalid email is entered	An invalid email address will prevent it being able to contact them.	bljnjk.jhb	Invalid	Error appears			
3	The Database integrity	If an invalid type of book ID is entered the program must recognise this and not allow it to be saved	Setting genreID to invalid TypeofBookID	Invalid	Error			
4	Does home page load	N/A	N/A	N/A	Loaded homepage			
5	Does list of genres appear when type	Fiction is one of the type of books meaning the list of genres should	Fiction is selected	Valid	List of genres to appear			

	of book is selected	appear when clicked						
6	Do the books appear when a genre is selected	Action and adventure is one of the genres belonging to fiction	Action and adventure is selected	Valid	List of action and adventure books to appear			
7	Search bar	There is a book called Harry Potter that already exists in the database	Harry Potter	Valid	List of Harry Potter books			
8	View basket	Clicking on the view basket button is the only way of viewing the basket	View basket clicked	Valid	Basket to appear			
9	Add to basket	Clicking on the add to basket button is the only way of adding a book	Add to basket clicked	Valid	The book to appear in the basket			
10	Update basket	The number of days that the user wants to rent for	Number of days	Valid	Number of days to change			
11	Delete from basket	The delete button is the only way to remove a	Clicking the delete button	Valid	The book to be removed			

		book from the basket			from basket			
12	Sign up page is displayed	Clicking the signup button is the only way for the user to enter the page	Sign up button is clicked	Valid	Signup page to be displayed			
13	Sign up successful	This will check to see if the user's info is saved to the user table	Fill out the details on the page	Valid	The website informs the user that it has been saved			
14	Add book page displayed	This will check if the page is able to load after the designated button is clicked	The add book button is clicked	Valid	The add book page should be displayed			
15	Book added	This will check to see if the book is saved to the database	Fill out the details on the page	Valid	Should be able to search for it after added			

Post Development Test Data

Test No.	Aspect being tested	Justification	Test Data	Type of test (valid, invalid, boundary)	Expected result	Actual result	Evidence (screenshot, screencast etc)	Actions needed (e.g. bug-fix and retest, ignore etc)
1	Does home page load	N/A	N/A	N/A	Loaded homepage			
2	Does list of genres appear when type of book is selected	Fiction is one of the type of books meaning the list of genres should appear when clicked	Fiction is selected	Valid	List of genres to appear			
3	View basket	Clicking on the view basket button is the only way of viewing the basket	View basket clicked	Valid	Basket to appear			
4	Add to basket	Clicking on the add to basket button is the only way of adding a book	Add to basket clicked	Valid	The book to appear in the basket			
5	Update basket	The number of days that the user wants to rent for	Number of days	Valid	Number of days to change			
6	Delete from basket	The delete button is the only way to	Clicking the delete button	Valid	The book to be removed			

		remove a book from the basket			from basket			
7	The website must allow the user to purchase a book.	This is the main feature my website is to do, if it is unable to do this, then my solution to the problem does not work.	Purchase Harry Potter	Valid	Email of confirmation			
8	Payment must be able to be done through PayPal	PayPal is the most commonly used software for people to use when buying online things, making it necessary that my solution can integrate with it.	Pay for book using PayPal	Valid	Payment successful			
9	Sort by genre	This method of searching through the database has been mentioned in my survey by the target market. It is also a common feature in many E-Commerce sites including the two I have investigated, eBay and Amazon.	Select comedy from fiction	Valid	List of comedy books to be shown			

10	Predictive search	Predictive search is also very common among E-Commerce websites. I came across this when investigating Amazon, and it is a very useful tool for users, especially if they are using a device which does not have a keyboard for them to type on.	Start typing in search box until books are suggested underneath it	Valid	Books to appear under search bar			
11	Suggestive search	Suggestive search is probably the most useful tool for searching, which is why one of my target market mentioned this in the survey. This is because it helps you find what you are looking for if your spelling is wrong or if your search criteria is a bit off.	When having searched for something, the website will then suggest some other books based on what you have searched	Valid	A list of suggested books to be shown			
12	Sign up and login area should be able to	The signup and login areas should be able to handle errors because	Type in an invalid email when	Invalid	Error should appear to tell user that			

	handle errors	if the user types in an invalid email address or a different piece of information, the website should be able to realise this and let the user know of the error. It should also be able to check that every piece of required information has been filled in correctly.	signing up		sign up has failed			
13	Sign up and login area should be able to handle errors	The signup and login areas should be able to handle errors because if the user types in an invalid email address or a different piece of information, the website should be able to realise this and let the user know of the error. It should also be able to check that every piece of required information has been filled in correctly.	Type in the wrong password when login in	Invalid	Error should appear to tell user that they have either entered the email or password in wrong			

14	The sign-up area must be able to check and save to the correct database	The sign-up area must be able to only access the customer database. This is so that when the user clicks the submit button their information will not be saved in the book database instead. It must also be able to search through the customer database to make sure that there are no existing databases too like the one the user is trying to make.	Sign up to website	Valid	A sign up successful message should appear and details should be in table			
15	The login area must be able to open the correct account	The login area must be able to open the customer database, check if the account exists, and then open for the user to use. This is so that no fake user may use the services of the website or pretend to be a different user. It must be able to open their account so that they may be	Login to the website	Valid	Login successful			

		able to see suggested books and be able to update details such as their Email addresses.						
16	Read list	This is a useful feature as it enables the user to keep track on what they want to read without the worry of forgetting one of them or trying to write them down.	Open read list and add a book	Valid	Successfully add a book to read list and view it.			

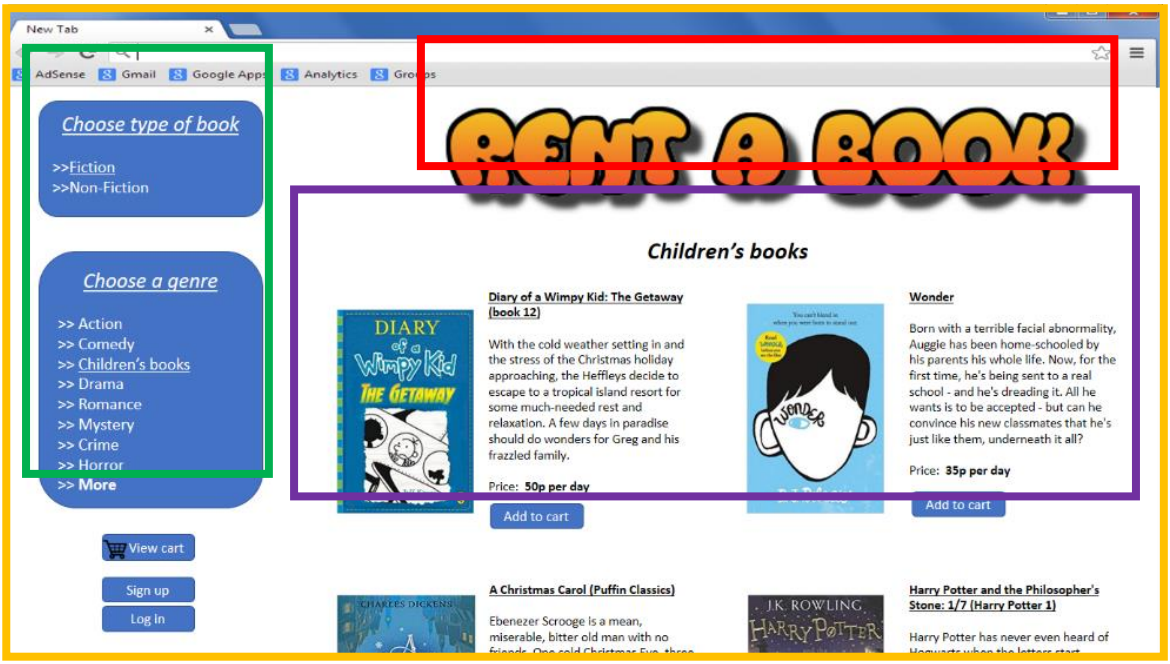
Developing the coded solution ("The Development Story")

Technology choices

I have decided to use visual basic.net for my language in order to make the solution. I have decided this because there are multiple tutorials I can learn from as the language has been around for a while and I will be able to take advantage of all the features provided by the .NET framework. Vb.net is also a fully object-oriented language. Along with this I will use ASP to allow me to build a dynamic and interactive website. The software integrated Development Environment application I'm going to use will Microsoft visual studios. This is because it has many useful features such as a form layout window and a debugger. These features will help me as it will enable me to step through my code line by line to find and fix errors, using the form layout window will make it a lot easier for me to design the website the way I want it.

Iteration 1 - Date 18/10/2017

Aims for this iteration



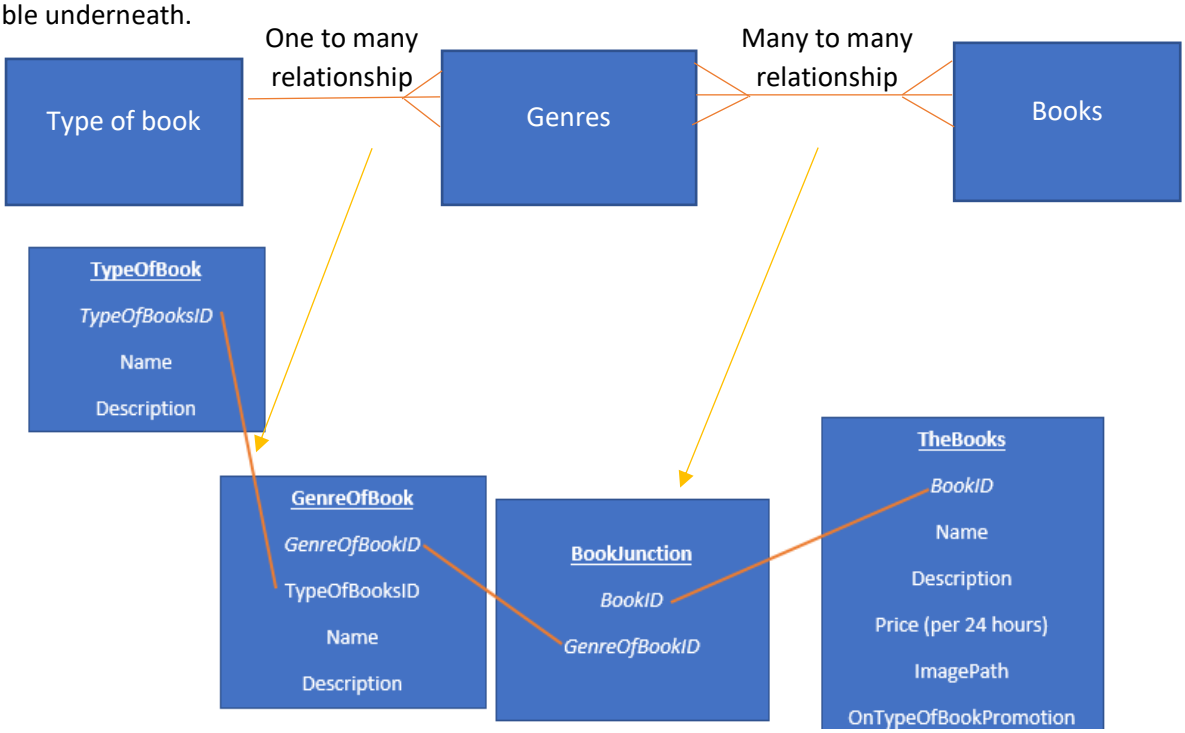
In this prototype, I aim to build the webpage and have the header displayed on it. I will then add the categories to the webpage with the genres associated with them to appear once clicked. This prototype will also have the specific books that belongs to the genres to be queried and shown on the website, like my design plan.

Functionality that the prototype will have

In order for this prototype to work it will need three tables containing the type of books, the genre of books and the books themselves. As there will be a many to many relationship between the genre of books and the books, I will need a junction table. The tables will then need to be populated and connected to the website. It will also need SQL stored procedures to query and get data to and from the different tables. The prototype will also need different user controls which can be reused around the website.

Diagram of the relationships between the tables in the database

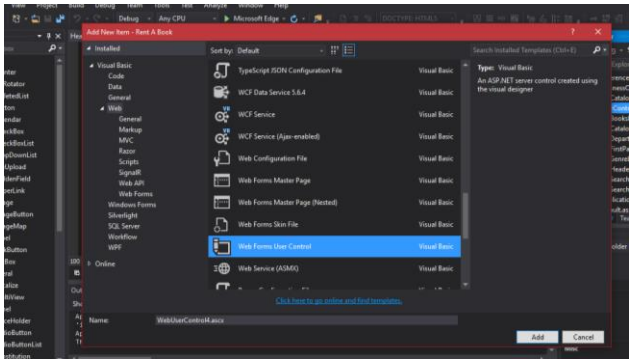
This entity- relationship diagram represents the relationship that the books have with the genres, and the relationship the genres have with the type of books. The relationship between the types of books and the genre of books is one to many, as there can be multiple genres belonging to one type of book. While the relationship between genres and books are many to many. This is because multiple genres can belong to one book and multiple books can belong to one genre. Because it is a many to many relationships is used, a junction table is needed, which is why there is a book junction table underneath.

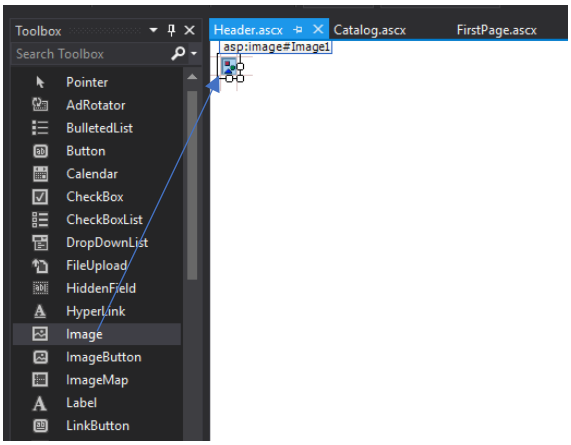


Annotated code screenshots with description

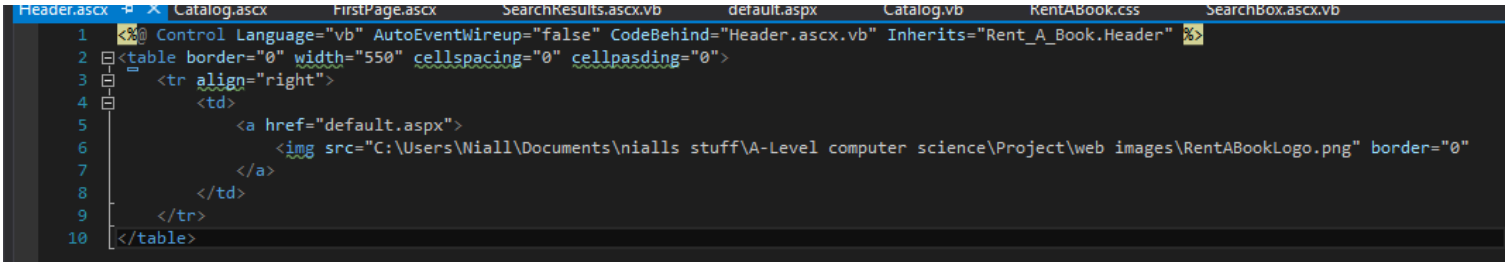
Making the homepage and header

Next, I made a UserControls folder to hold a new item in. The new item I added was a web forms user control called Header.ascx. I then opened this new file in design view and added an image control from toolbox onto the page.

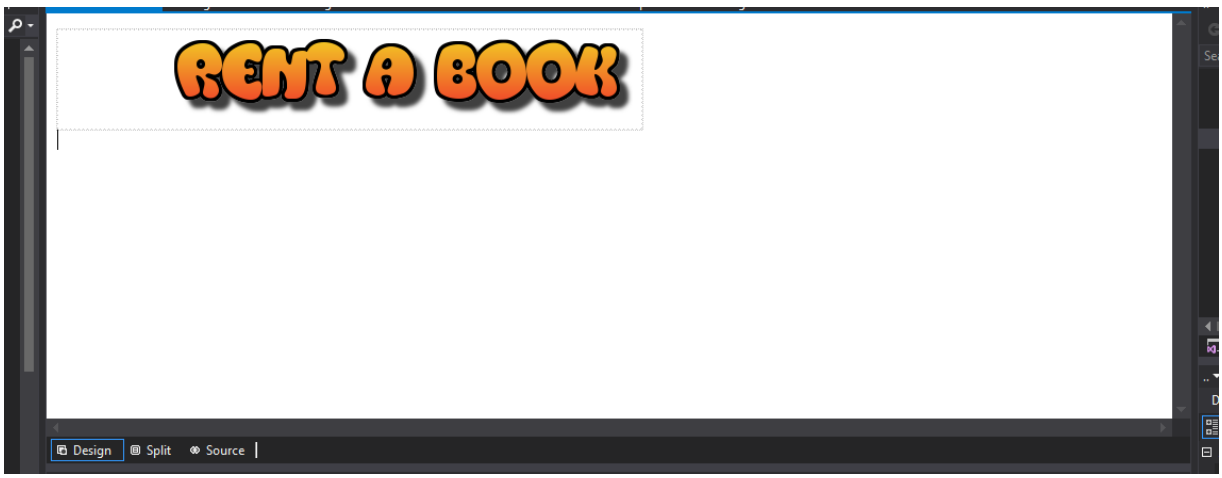


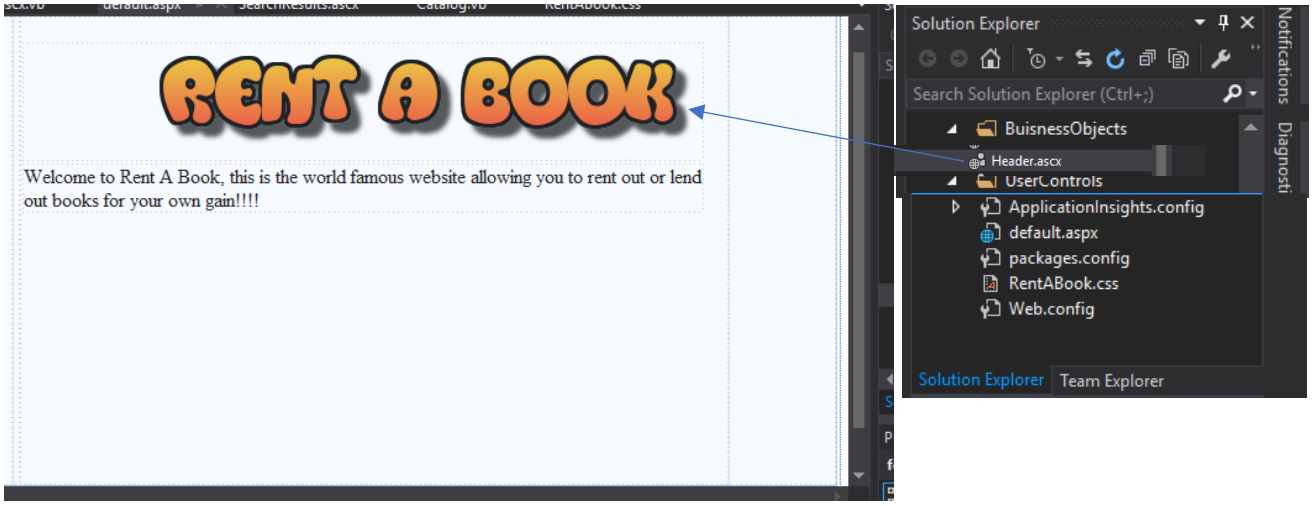


After this I needed to go back to source view and change the img src line of code to the location of the .png file the Rent A Book title is.

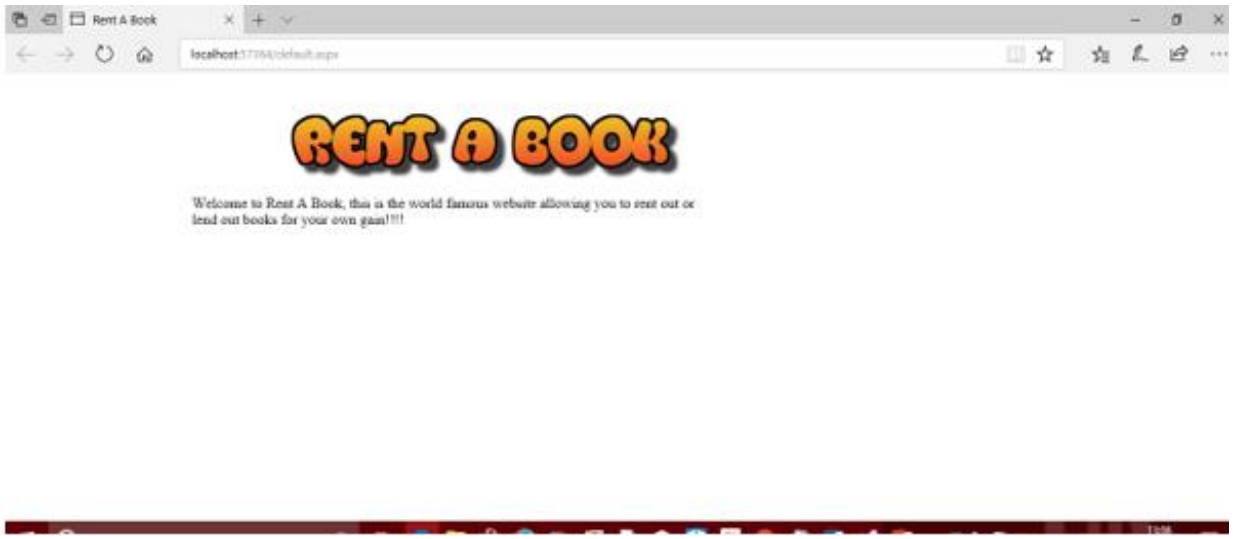


By doing this it adds the image to the file as shown in design view below:



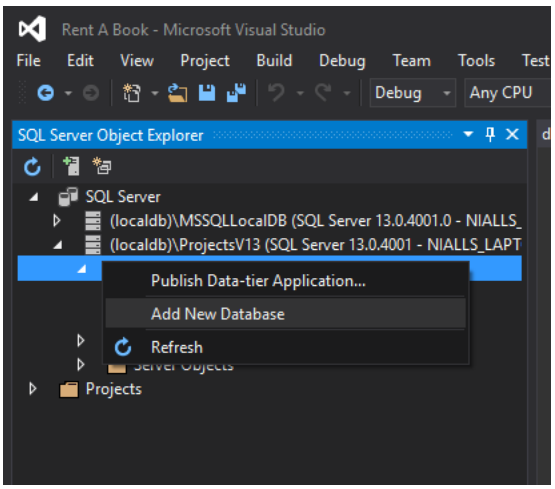


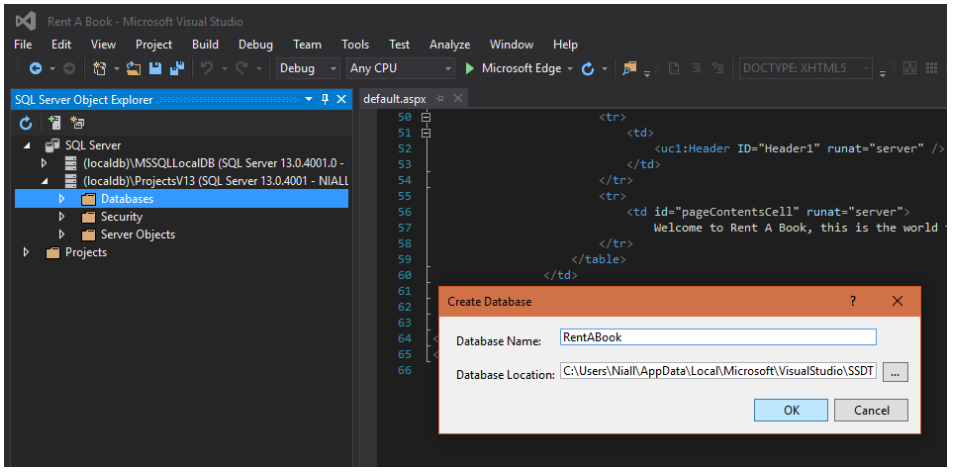
In order to add the user control called header.ascx I went onto design view in default.aspx and dragged and dropped the header.ascx file onto the page. When executed in Microsoft edge the website's home page looks like this:



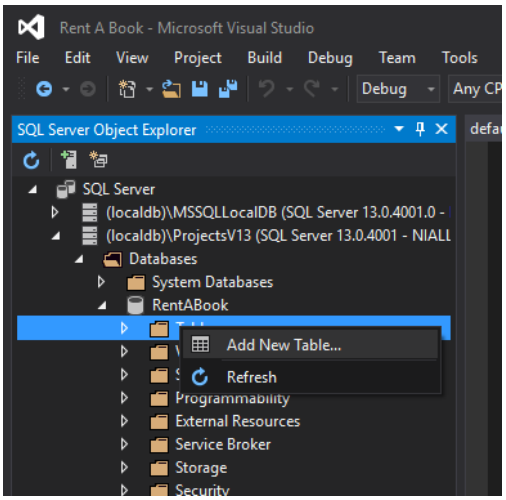
Making the Rent a Book database and the types, genres, books tables

To create a new database, I first went to SQL server object explorer and right clicked on databases. After that I typed the name for my new database, RentABook, and choose its location.

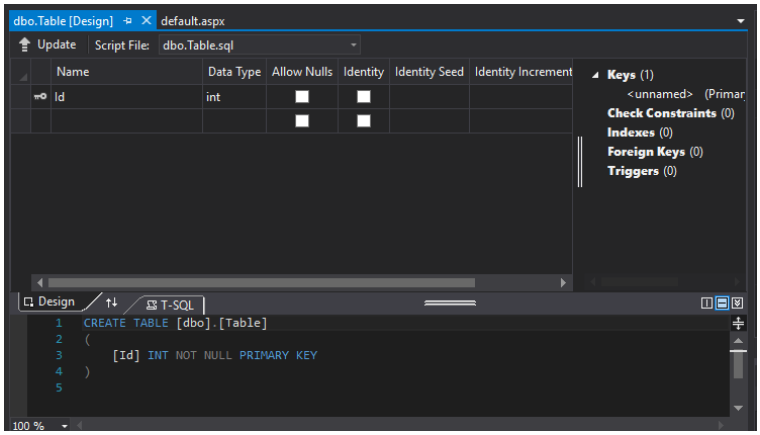


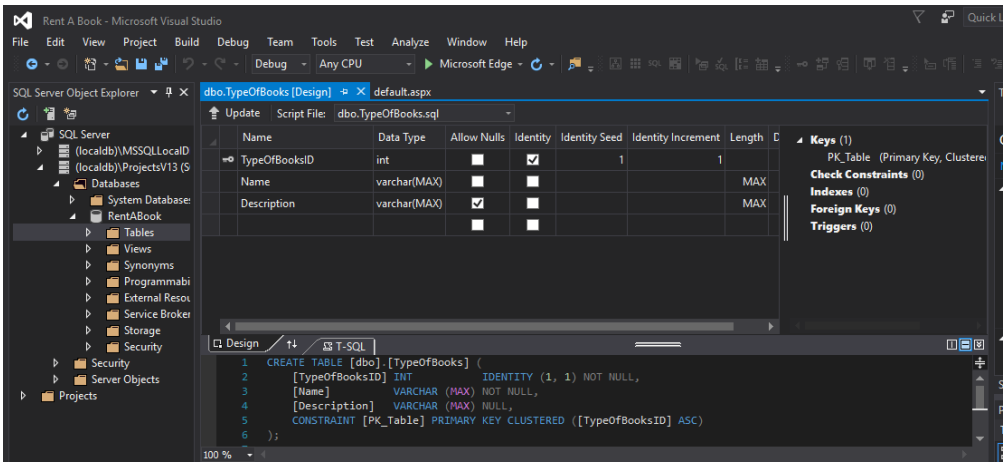


Now that the database is made I am able to build the three tables. The first table I made was the type of book table, which will hold fiction and non-fiction along with a definition on both of them. To do this I opened the RentABook database, right clicked on tables and choose Add New Table.



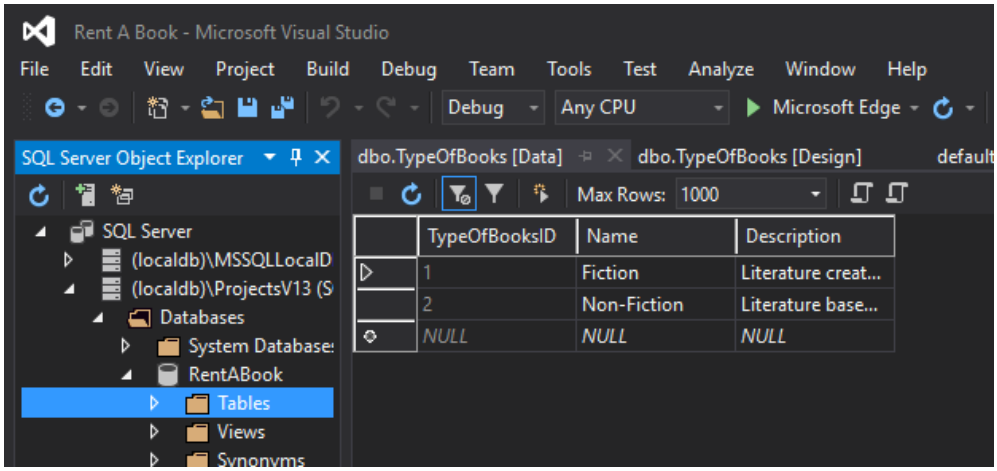
By doing this I made a new blank table.





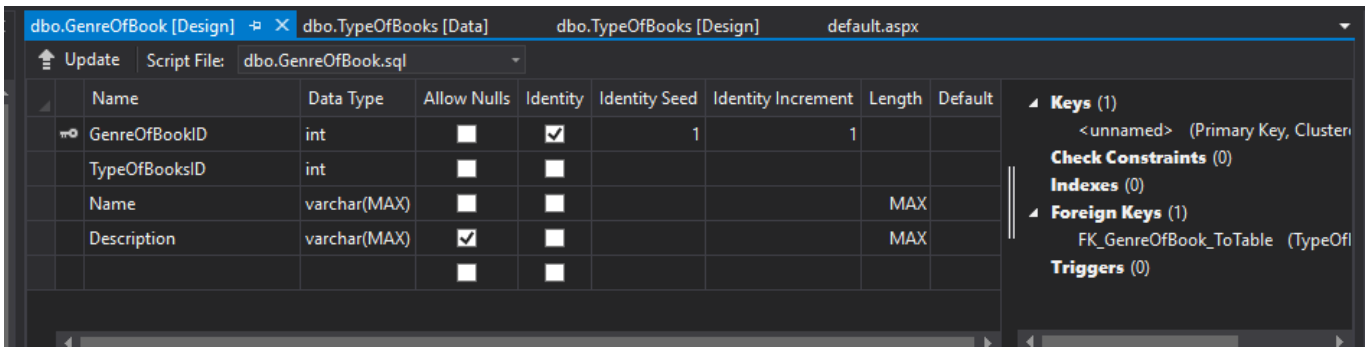
Next, I added the names of the columns (TypeOfBooksID, Name and Description) and set TypeOfBooksID to be an identity and primary key. This means when I populate the table, each record will have a unique identifier called TypeOfBooksID. This also meant that I needed to set TypeOfBooksID to be a int datatype, while Name and Description can be both varchar.

After the table has been made I went onto the task of populating it. To do this I right clicked the table and clicked on view data. I then populated it with the following data.



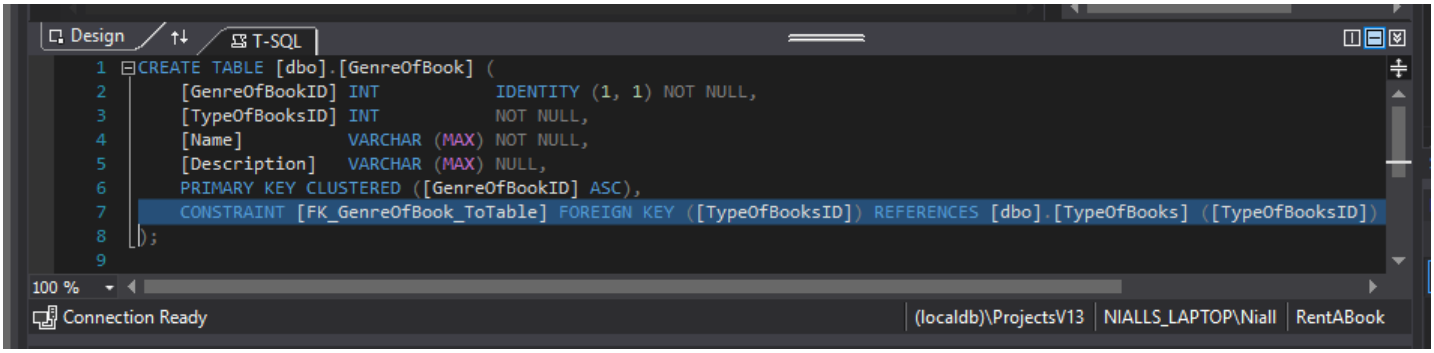
As you can see the identity column (TypeOfBooksID) has automatically incremented by one for each record.

The next table I made was the genres table. This table was a little different to the previous table as this one also contained a foreign key. It needed a foreign key to create a physical relationship between two tables. The foreign key in this table was TypeOfBooksID, this meant that the GenreOfBook table will have a relationship with the TypeOfBooks table.



However, this table still had the same Name and Description columns as the TypeOfBooks table. It also has an identity and primary key column, but this tables one is called GenreOfBookID.

In order to make the TypeOfBooksID column a foreign key I needed to add another line in the T-SQL area.



After making the table I then populated it with the necessary data.

dbo.GenreOfBook [Data] | dbo.GenreOfBook [Design] | dbo.TypeOfBooks

Max Rows: 1000

GenreOfBookID	TypeOfBooksID	Name	Description
1	1	Science fiction	NULL
2	1	Drama	NULL
3	1	Action and Adv...	NULL
4	1	Romance	NULL
5	1	Mystery	NULL
6	1	Horror	NULL
7	2	Self help	NULL
8	2	Health	NULL
9	2	Guide	NULL
10	2	Travel	NULL
11	1	Children's	NULL
12	2	Religion, Spiritu...	NULL
13	2	Science	NULL
14	2	History	NULL
15	2	Math	NULL
16	2	Encyclopedias	NULL
17	2	Dictionaries	NULL
18	1	Comics	NULL

Connection Ready

As you can see, because GenreOfBookID is the unique identifier, it has given each record a unique ID. The name column contains all the different genres, fiction and non-fiction, with description which will hold a sentence or two explaining what the genre is.

In the TypeOfBooksID column, you can see that each record either has a 1 or a 2 inside it. This is because it is a foreign key, meaning it represents the unique IDs of the records in the TypeOfBooks table. This means that if the record contains a 1 in the TypeOfBooksID column, such as records 3 and eleven, it means that the genre belongs to a fictional type of books. If it had a 2 in the column TypeOfBooksID instead, it would mean that the genre belongs to a non-fictional type of books, such as records 7 and sixteen.

The books table contains the most data out of the three tables. This is because it will hold thousands of books (when the website is used by lots of people) and for each book/record it will hold its book ID, Name, Description, Price Per Day, Image Path and whether it will be on type of book promotion or genre of book promotion.

dbo.TheBooks [Design] | dbo.GenreOfBook [Data] | dbo.GenreOfBook [Design] | dbo.TypeOfBooks [Data] | dbo.TypeOfBooks [Design]

Update | Script File: dbo.TheBooks.sql

Name	Data Type	Allow Nulls	Identity	Identity Seed	Identity Increment	Length	Default
BookId	int	☐	☑	1	1		
Name	varchar(MAX)	☐	☐			MAX	
Description	varchar(MAX)	☑	☐			MAX	
PricePerDay	money	☐	☐				
ImagePath	varchar(MAX)	☑	☐			MAX	
OnTypeOfBookPromotion	bit	☐	☐				((0))
OnGenreOfBookPromotion	bit	☐	☐				((0))
		☐	☐				

Keys (1)
<unnamed> (Primary Key, Cluste
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

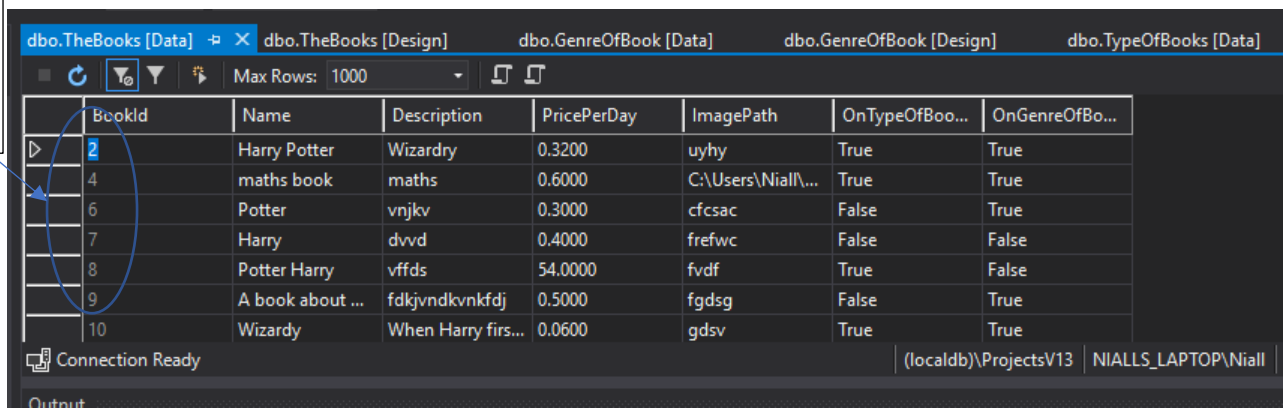
Design | T-SQL

```
CREATE TABLE [dbo].[TheBooks] (
    [BookId]                INT             IDENTITY (1, 1) NOT NULL,
    [Name]                  VARCHAR (MAX)   NOT NULL,
    [Description]           VARCHAR (MAX)   NULL,
    [PricePerDay]           MONEY           NOT NULL,
    [ImagePath]             VARCHAR (MAX)   NULL,
    [OnTypeOfBookPromotion] BIT             DEFAULT ((0)) NOT NULL,
    [OnGenreOfBookPromotion] BIT            DEFAULT ((0)) NOT NULL,
    PRIMARY KEY CLUSTERED ([BookId] ASC)
);
```

I set the BookID column to be the identity and primary key so that each book will have a unique ID even if the same book is put in twice. The name, description and ImagePath columns are put a varchar Data type with a length of maximum, meaning the user can put in an as big of a description or name they want. PricePerDay is set to data type money as it will be holding the value of the books, while OnTypeOfBookPromotion and OnGenreOfBookPromotion have bit for data type as it will either contain True or False.

After the table being made I then populated it with a few books.

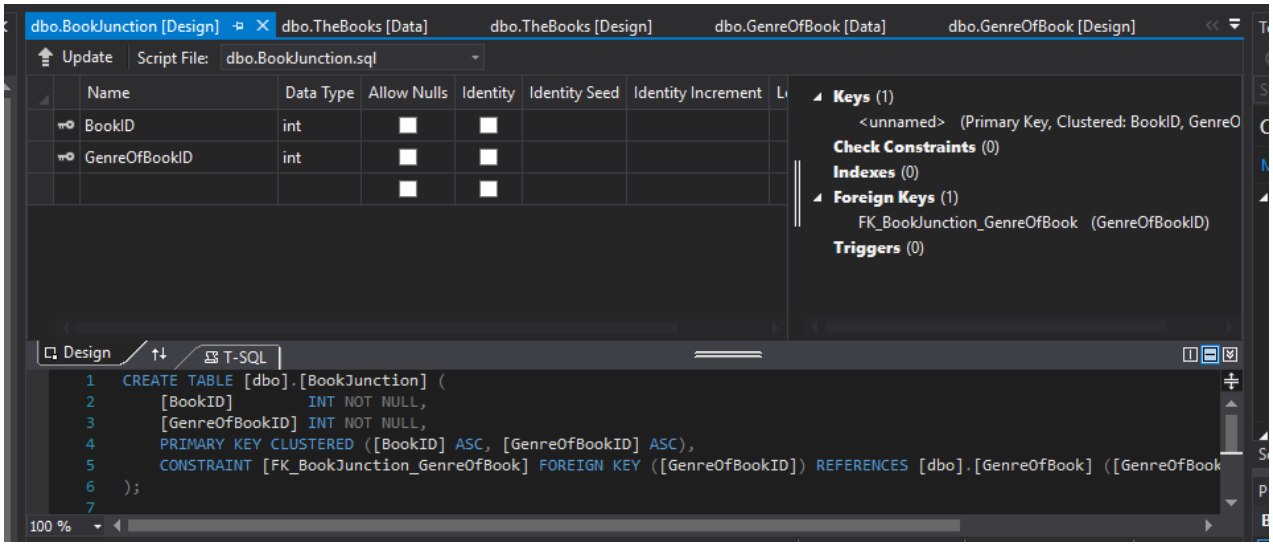
This was created by me accidently



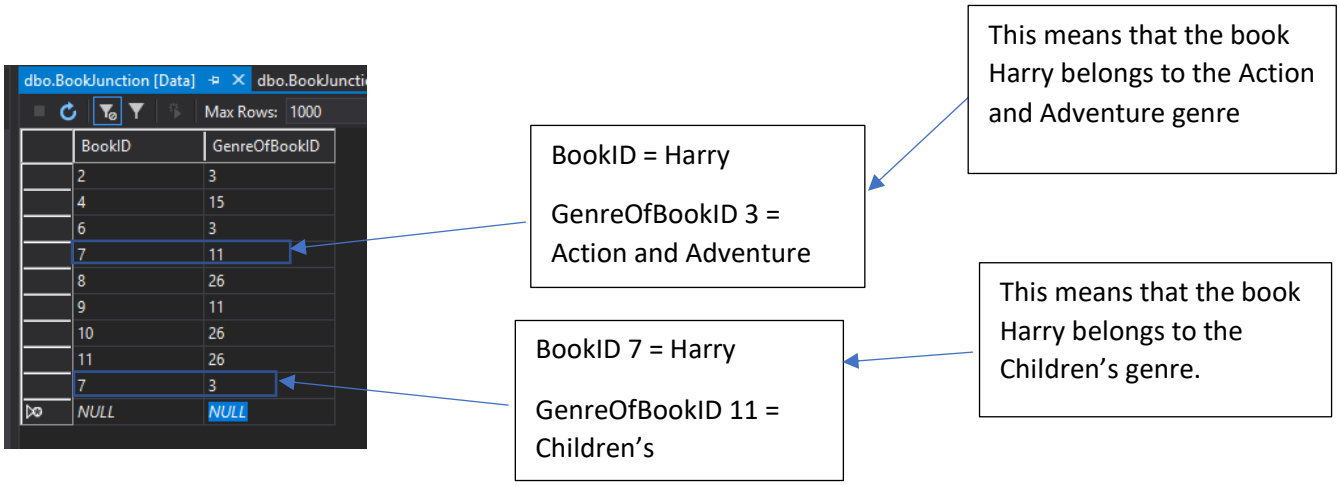
The BookID column holds the unique ID of each book, while the Name, Description, PricePerDay and ImagePath holds the data specific to the book. The OnTypeOfBookPromotion and OnGenreOfBookPromotion decides whether it is shown when the user selects the type or genre of the book they are looking for.

This table will also require a relationship with the GenreOfBook table. However, unlike the one to many relationship the GenreOfBook table had with the TypeOfBooks table, TheBooks table will require a many to many relationship. This is because one book/record can belong to multiple genres. In order to make the many to many relationship I need another table called BookJunction.

The BookJunction table will enable the many to many relationship by containing the primary keys of both tables, TheBooks and GenreOfBook, as foreign keys. This means that each table will have a one to many relationship with the BookJunction table, creating a many to many relationship with each other.

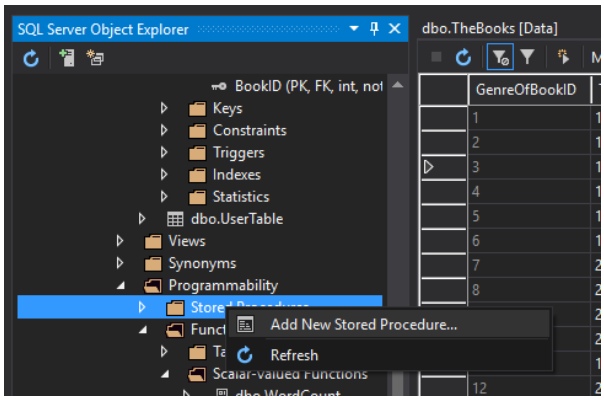


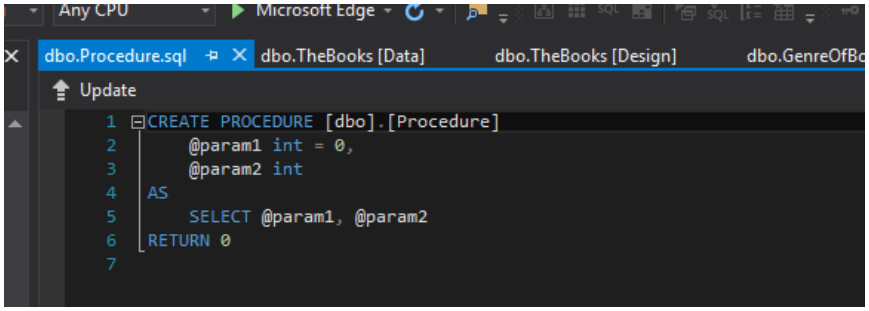
After building the table I then populated it with data. In the BookID column I added the unique ID of the book I wanted to place. Then under the GenreOfBookID column I put down the unique ID of the genre corresponding to the book. Also, because I have not set any of the columns to be a unique identity the same BookIDs and GenreOfBookIDs can be referred to repeatedly.



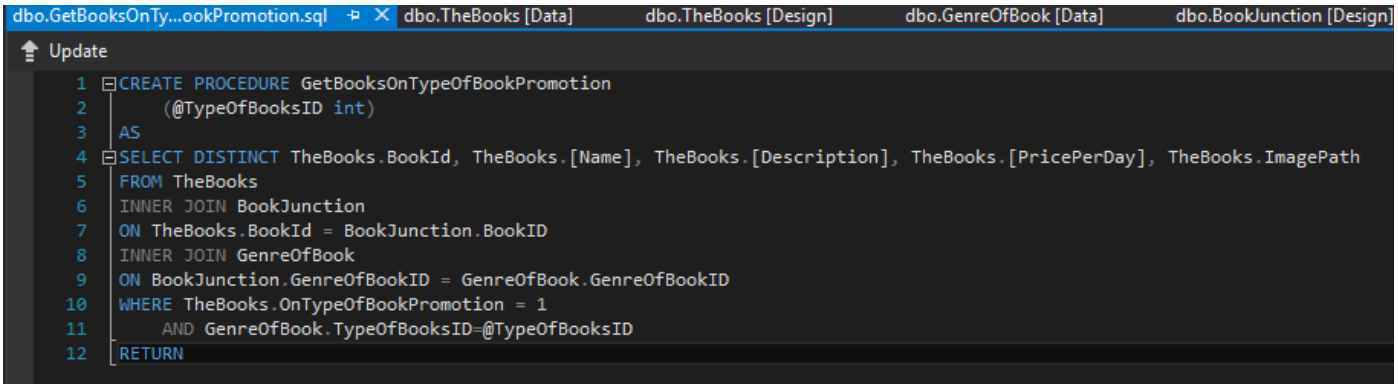
Making the SQL stored procedures

The first SQL stored procedure I made was the GetBooksOnTypeOfBookPromotion procedure. This procedure is meant to return all the books that have 'true' in the OnTypeOfBookPromotion column in TheBooks table. In order to make this stored procedure, I first went onto my RentABook database, opened the programmability folder and right clicked the stored procedures folder. I then clicked on Add New Stored Procedure, creating the new procedure.



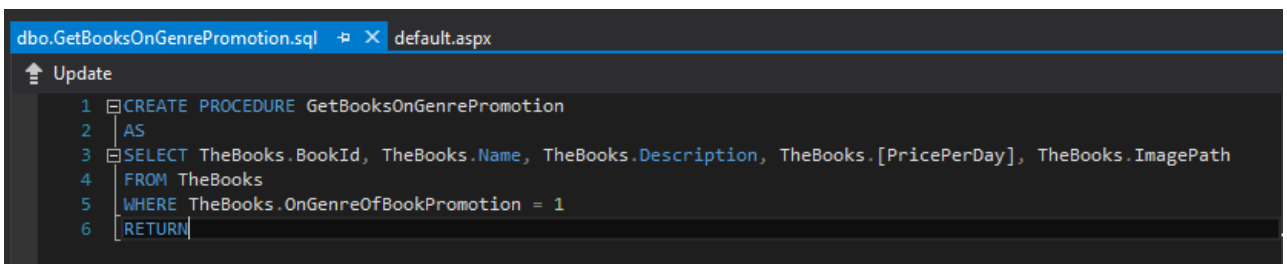


This procedure will take in a parameter called @TypeOfBooksID which is an integer, it will only contain a 1 or a 2 representing Fiction and non-fiction respectively. On line 4 I used the DISTINCT statement with the SELECT statement. If I did not use the DISTINCT statement the procedure will still return all the books for the specific promotion,



but because a book can belong to multiple genres under the same type of book (fiction or non-fiction), it would be repeated. The DISTINCT statement prevents this error by making sure that a single book ID is not repeated, preventing the same book being repeated on the type of book promotion pages.

The second SQL stored procedure I made was the GetBooksOnGenrePromotion procedure. This procedure will return all the books that have been selected to be shown if they are set to 1 in the column OnGenreOfBookPromotion on the TheBooks table.



This SQL procedure is much smaller than the last one as all it needed to do was to return all the books with a value of 1 in a specific column, whereas the other one had to check multiple tables.

The next two procedures are used to output the type of book name (fiction or non- fiction) and the description of it, such as the name “Fiction” and its description “Literature created from the imagination. Mysteries, science fiction, romance, fantasy, chick lit, crime thrillers are all fiction

```
dbo.GenreOfBook [Data]    dbo.TypeOfBooks [Data]    dbo.GetTypeOfBooksD
Update
CREATE PROCEDURE GetTypeOfBooksDetails
    (@TypeOfBooksID Int,
    @Name varchar(MAX) OUTPUT,
    @Description varchar(MAX) OUTPUT)
AS
SELECT @Name=[Name], @Description=[Description]
FROM TypeOfBooks
WHERE TypeOfBooksID = @TypeOfBooksID
RETURN
```

```
dbo.GetGenreOfBookDetails.sql    dbo.GenreOfBook [Data]    dbo.TypeOfBooks [Data]
Update
CREATE PROCEDURE GetGenreOfBookDetails
    (@GenreOfBookID int,
    @Name varchar(MAX) OUTPUT,
    @Description varchar(max) OUTPUT)
AS
SELECT
    @Name = [Name],
    @Description = [Description]
FROM GenreOfBook
WHERE GenreOfBookID = @GenreOfBookID
RETURN
```

genres.” While the other is used to output the same thing but for genres, genre name: “Comedy” description: “Refers to any discourse or work generally intended to be humorous or amusing by inducing laughter.”

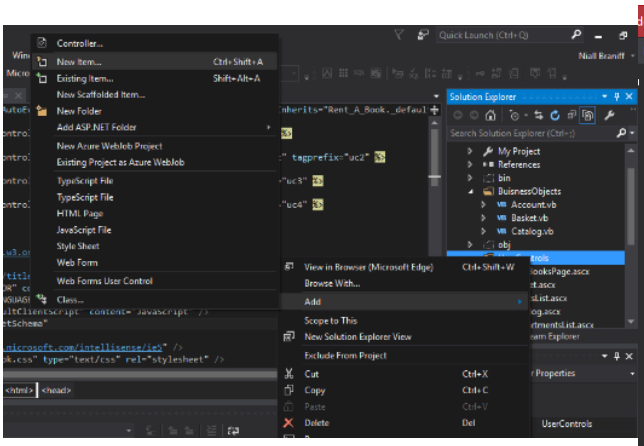
This procedure is used to get the books related to a specific genre. It does this by joining the two tables, TheBooks and the BookJunction tables. This means it is able to find and select the books/

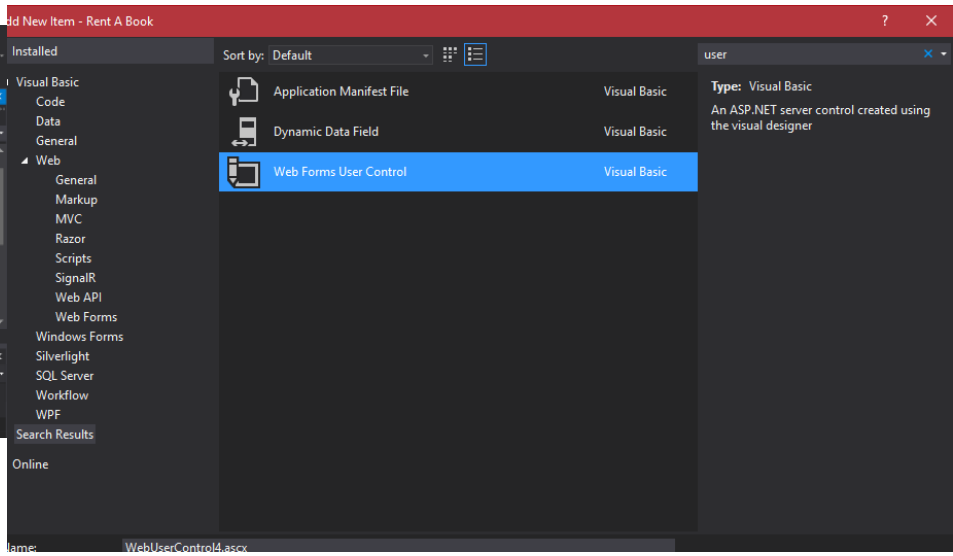
```
dbo.GetBooksInGenre.sql    dbo.GetGenreOfBookDetails.sql    dbo.GenreOfBook [Data]    dbo.TypeOfBooks [Data]    dbo.GetTypeOfBooksDetails.sql
Update
CREATE PROCEDURE GetBooksInGenre
    (@GenreOfBookID int)
AS
SELECT TheBooks.BookID, TheBooks.[Name], TheBooks.[Description], TheBooks.[PricePerDay], TheBooks.ImagePath, TheBooks.OnTypeOfBookPromotion, TheBooks.OnGenre
FROM TheBooks INNER JOIN BookJunction
ON TheBooks.BookId = BookJunction.BookID
WHERE BookJunction.GenreOfBookID = @GenreOfBookID
RETURN
```

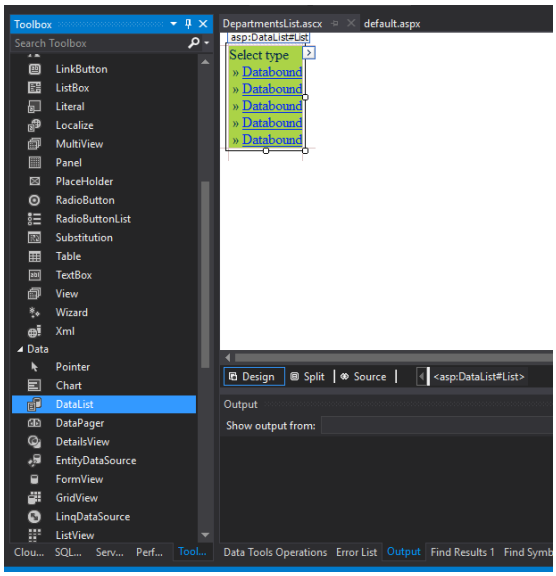
records where the genre of book IDs are the same giving the book’s name, description, price per day and back as a result.

Displaying the books under the selected type and genre of books

After making all the necessary procedures I then went onto creating the user controls for them. The first control I made was for displaying the list of different types of books. To do this I right clicked on the UserControls file I made and clicked on add and then on new item. Next, I searched for Web Forms User Control and named it departmentList.ascx

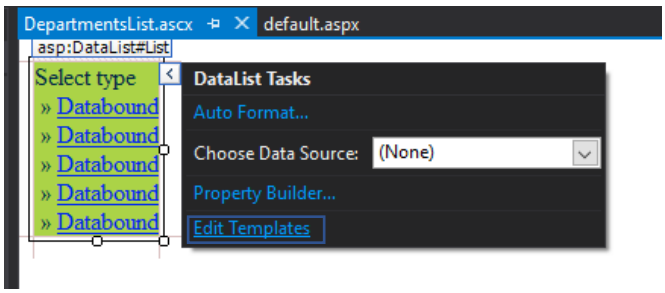




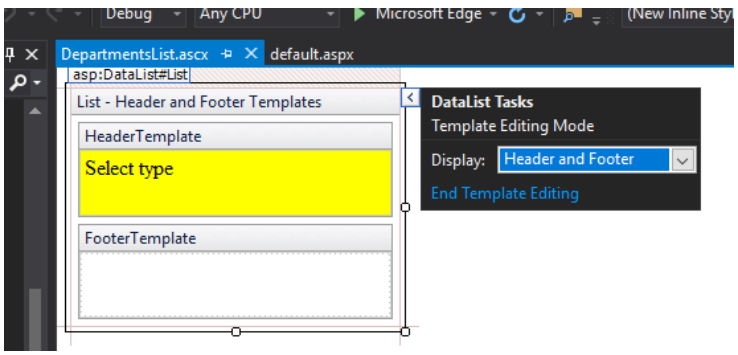


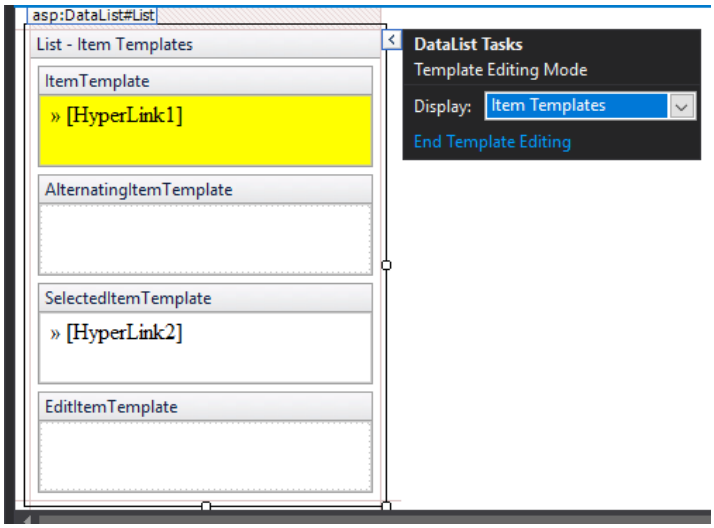
I then went onto design view and dragged and dropped Data List from toolbox.

Next, I went onto Edit Templates where I added the text



"Select Type" in to the header section of header and footer template. I also added two hyperlinks, one in item template and the other in selected item template.





I then set HyperLink1 CSS class to BookTypeUnselected and HyperLink2 to BookTypeSelected.

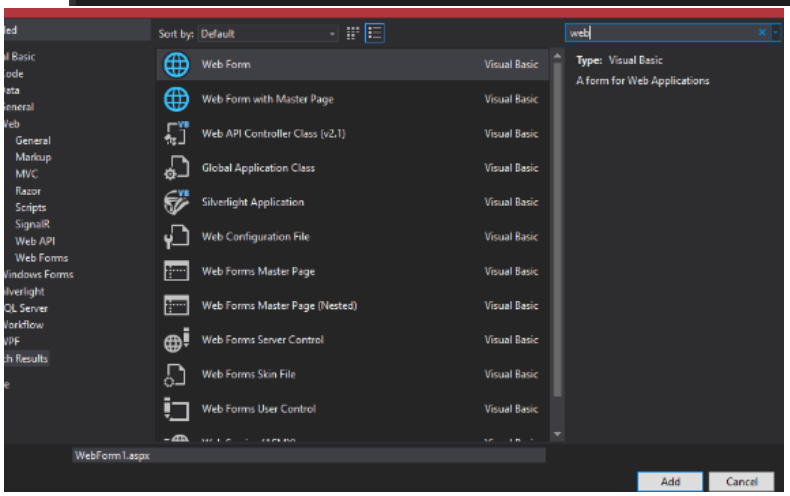
After that I made a CSS file so that I can edit the appearance of the website. To do this I right clicked on RentABook in solution explorer and clicked on add and then on stylesheet. I then named it to RentABook.css. After that I edited the code to this.

```
ntABook.css  DepartmentsList.ascx  default.aspx
1  .BookTypeUnselected
2  {
3      color: blue;
4      font-family: Comic Sans MS;
5      text-decoration: none;
6      font-size: 14px;
7      font-weight: bold;
8      line-height: 25px;
9  }
10
11 .BookTypeUnselected:hover
12 {
13     color: red;
14 }
15
16 .BookTypeSelected
17 {
18     color: green;
19     font-family: Comic Sans MS;
20     text-decoration: none;
21     font-size: 14px;
22     font-weight: bold;
23     line-height: 25px;
```

the 'hover' command after the title .BookTypeUnselected meant that when the user moved their mouse over the hyperlink, the hyperlink will change to the colour red.

I then right clicked departmentList.ascx and clicked on view code to see the behind code. I then changed it to this.

```
epartmentsList.ascx.vb*  RentABook.css  DepartmentsList.ascx  default.aspx
Rent A Book  (DepartmentsList Events)  Load
1  Public Class DepartmentsList
2      Inherits System.Web.UI.UserControl
3
4      Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
5          'The departmentIndex parameter is added to the query string when a department link is clicked.
6          'This is because that when the page is reloaded, the DataList forgets which link was clicked.
7          Dim listIndex As String = Request.QueryString("departmentIndex")
8          'If listIndex has a value, the user has clicked on a type of book
9          If Not listIndex Is Nothing Then
10             List.SelectedIndex = CInt(listIndex)
11         End If
12         'Get departments returns a SqlDataReader object that has two fields: TypeOfBookID and Name.
13         'These fields are read in the SelectedItemTemplate and ItemTemplate of the DataList
14         List.DataSource = Catalog.GetBookTypes()
15         List.DataBind()
16     End Sub
17
18 End Class
```



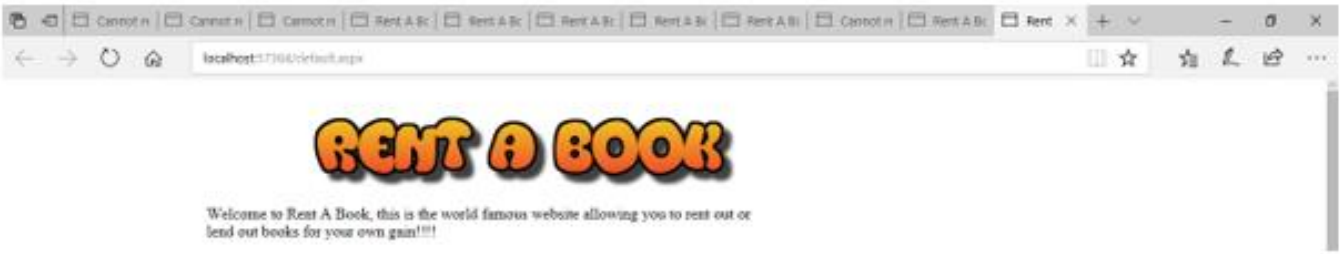
After that I made a catalog.vb file by right clicking and adding a Web form and calling it catalog.vb.

Catalog.vb | DepartmentsList.ascx.vb* | RentABook.css | DepartmentsList.ascx | default.aspx

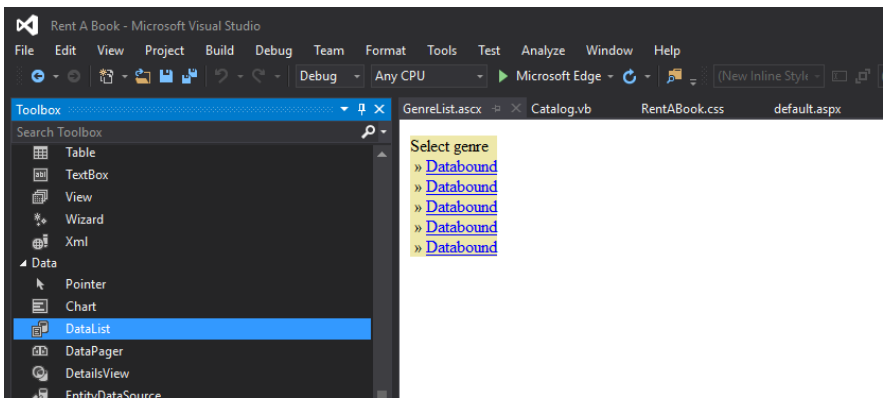
Rent A Book | TypeOfBookDetails

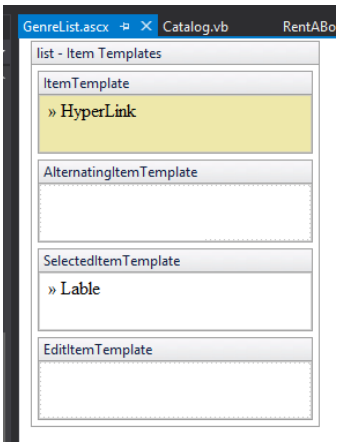
```
12
13  Public Class Catalog
14      Public Shared Function GetBookTypes() As SqlDataReader
15          ' Create the connection object
16          Dim connection As New SqlConnection(connectionString)
17          ' Create and initialize the command object
18
19          Dim command As New SqlCommand("GetTypeOfBooks", connection)
20          command.CommandType = CommandType.StoredProcedure
21          ' Open the connection
22          connection.Open()
23          ' Return a SqlDataReader to the calling function
24          Return command.ExecuteReader(CommandBehavior.CloseConnection)
25      End Function
26
27      Private Shared ReadOnly Property connectionString() As String
28          Get
29              ' Return ConfigurationSettings.AppSettings("ConnectionString")
30              Return System.Configuration.ConfigurationManager.AppSettings("ConnectionString")
31
32          End Get
33      End Property
```

I then made a class called Catalog which contained a function and a read only property. The read only property is used to help create a connection to the database. The function however is called GetBookTypes() and was called by the departmentList.ascx.vb file (catalog.GetBookTypes()). This function is used to execute the SQL stored procedure called GetTypeOfBook which will return the names in TypeOfBooks table to be displayed on the website. I then went back onto default.ascx and then dragged and dropped the user control onto it in design view.



Making the genre list was very much similar to the department list, as I needed to drag and drop a data list from toolbox onto a new web user control form called GenreList.ascx.





I also added a hyperlink and label to the item template and selected

item template like the department list one.

For the code behind file I have edited it to this, so that it is able to check if the user has selected a genre or not. If it has not it will call the function in the Catalog class called GetGenresInTypeOfBook

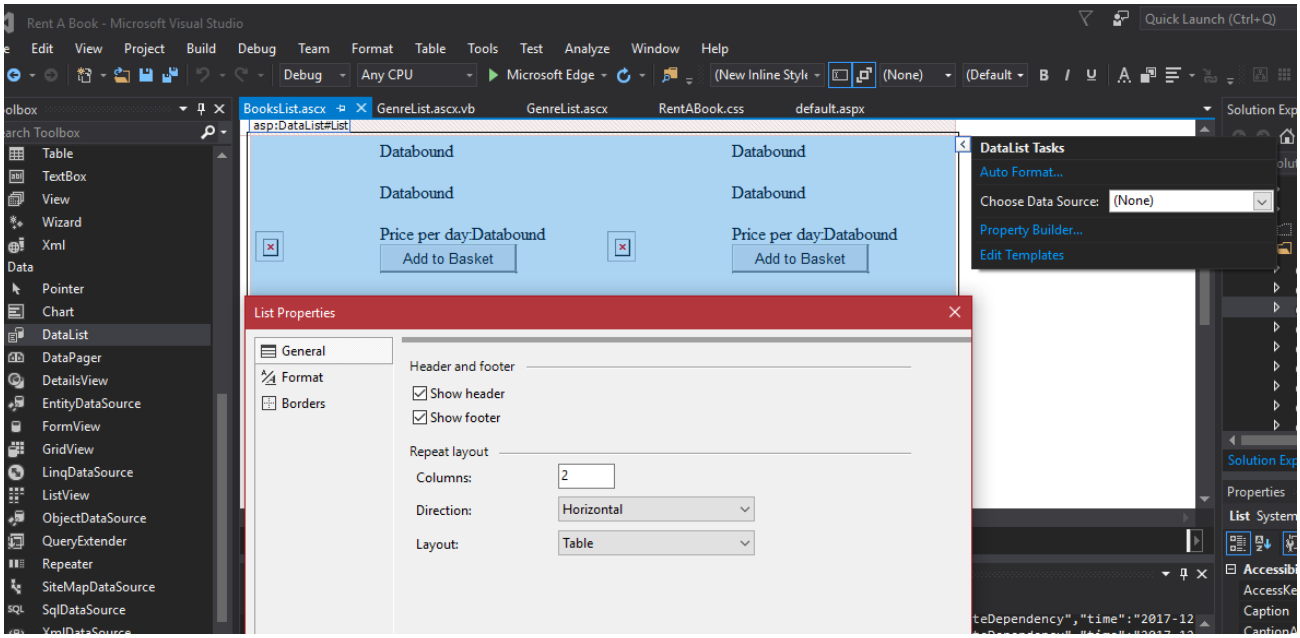
```
GenreList.ascx.vb | GenreList.ascx | Catalog.vb | RentABook.css | default.aspx
Rent A Book | (WebUserControl1 Events) | Load
1   Public Class WebUserControl1
2       Inherits System.Web.UI.UserControl
3
4       Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
5           Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
6           If Not TypeOfBooksID Is Nothing Then
7               Dim listIndex As String = Request.QueryString("GenreIndex")
8               If Not listIndex Is Nothing Then
9                   list.SelectedIndex = CInt(listIndex)
10              End If
11              list.DataSource = Catalog.GetGenresInTypeOfBook(TypeOfBooksID)
12              list.DataBind()
13          End If
14      End Sub
15
16  End Class
```

with a parameter of the type of book ID.

```
Public Shared Function GetGenresInTypeOfBook(ByVal TypeOfBooksId As String) As SqlDataReader
    ' Create the connection object
    Dim connection As New SqlConnection(connectionString)
    ' Create and initialize the command object
    Dim command As New SqlCommand("GetGenreFromTypeOfBooks", connection)
    command.CommandType = CommandType.StoredProcedure
    ' Add an input parameter and supply a value for it
    command.Parameters.Add("@TypeOfBooksID", SqlDbType.Int, 4)
    command.Parameters("@TypeOfBooksID").Value = TypeOfBooksId
    Try
        'Open the connection
        connection.Open()
        ' Return an SqlDataReader to the calling function
        Return command.ExecuteReader(CommandBehavior.CloseConnection)
    Catch ex As Exception
        ' Close the connection and throw the exception
        connection.Close()
        Throw ex
    End Try
End Function
```

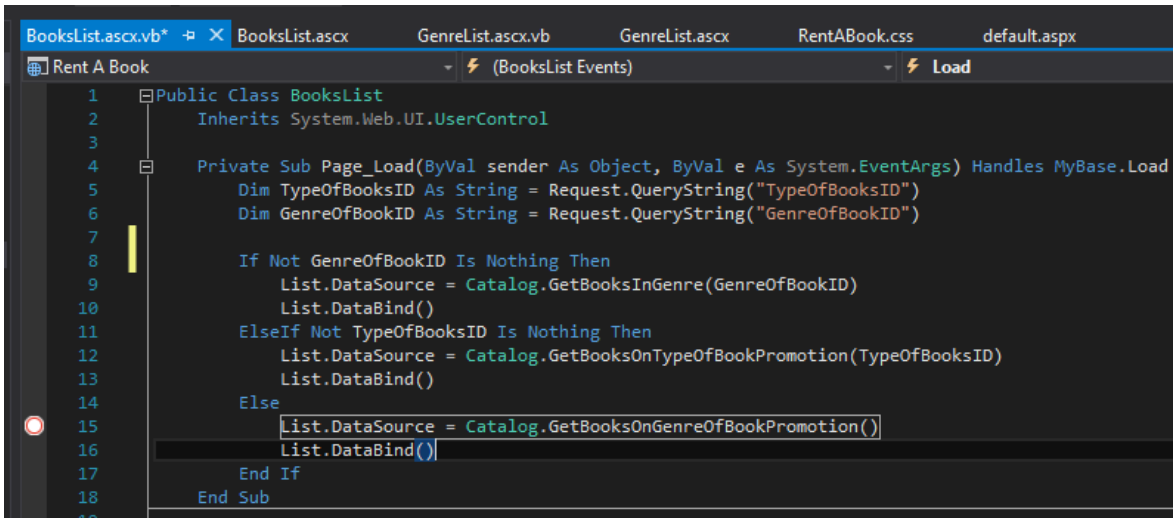
I then added another function to the Catalog class called GetGenresInTypeOfBook. This function calls the SQL stored procedure called GetGenresFromTyoeOfBooks and passes in the Type of book ID, enabling the function to return a list of the genres that belong to either fiction or non-fiction to be displayed on the website. I then went back onto default.ascx and then dragged and dropped the user control onto it in design view.

After that I made the final user control of this prototype. The BookList. To make this user control, I did the same as before by right clicking, pressing add and then selecting web user control form and



then changing its name to BookList.ascx. I then added a data list from toolbox, but then I went to property builder and set it to have two columns. This allows the books to be displayed in rows of two on the webpage.

Next, I went to the code behind file and edited it as follows. I have edited it so that it will first check to see if one of the genres have been selected. If it has been selected it will call the function

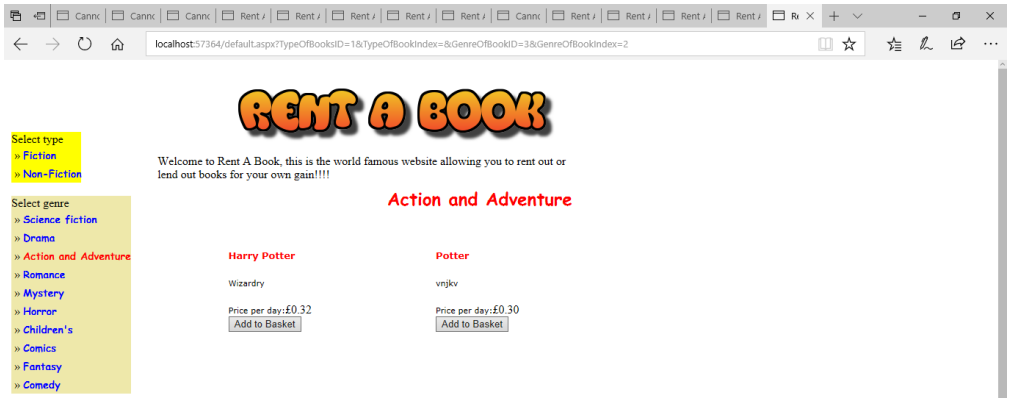


GetBooksInGenre from the Catalog class and then bind the results with the list. If none of the genres have been selected it will then check to see if one of the types of books have been selected, if so it will call the function GetBooksOnTypeOfBookPromotion from the Catalog class and binds it to the list. Otherwise it will call the function GetBooksOnGenreOfBookPromotion from the Catalog class.

I then went back onto Catalog.vb and added the three extra functions, GetBooksInGenre,

```
15
16      Public Shared Function GetBooksInGenre(ByVal GenreOfBooksId As String) As SqlDataReader
17          ' Create the connection object
18          Dim connection As New SqlConnection(connectionString)
19          ' Create and initialize the command object
20          Dim command As New SqlCommand("GetBooksInGenre", connection)
21          command.CommandType = CommandType.StoredProcedure
22          ' Add an input parameter and supply a value for it
23          command.Parameters.Add("@GenreOfBookID", SqlDbType.Int, 4)
24          command.Parameters("@GenreOfBookID").Value = GenreOfBooksId
25          Try
26              'Open the connection
27              connection.Open()
28              ' Return an SqlDataReader to the calling function
29              Return command.ExecuteReader(CommandBehavior.CloseConnection)
30          Catch ex As Exception
31              ' Close the connection and throw the exception
32              connection.Close()
33              Throw ex
34          End Try
35      End Function
```

GetBooksOnTypeOfBookPromotion and GetBooksOnGenreOfBookPromotion. The GetBookInGenre function executes the SQL stored procedure called GetBooksInGenre with the parameter GenreOgBookID. The procedure will then return the book's, that belongs to the specific genre selected, name, description and price per day. These are then returned out of the function and into the list where it can be displayed to the user.

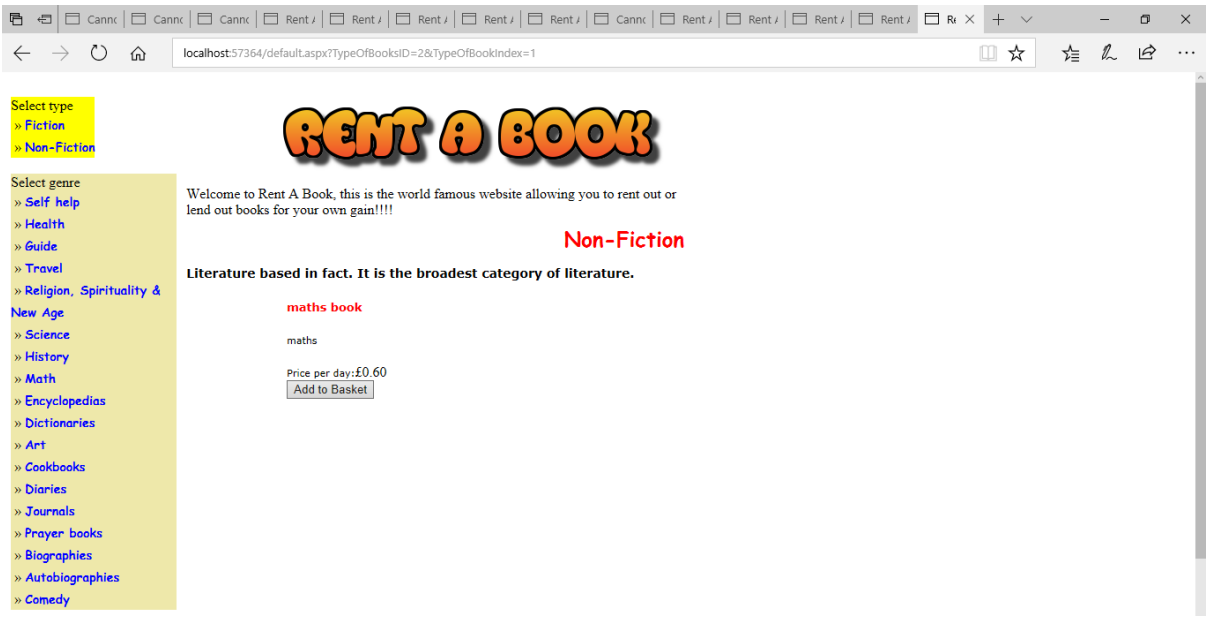


The GetBookdOnTypeOfBookPromotion function calls the GetBooksOnTypeOfBookPromotion that will return the book's, that belongs to the specific genres that belongs to the type of book

```
6
7       Public Shared Function GetBooksOnTypeOfBookPromotion(ByVal TypeOfBooksID As String) As SqlDataReader
8           ' Create the connection object
9           Dim connection As New SqlConnection(connectionString)
0           ' Create and initialize the command object
1           Dim command As New SqlCommand("GetBooksOnTypeOfBookPromotion", connection)
2           command.CommandType = CommandType.StoredProcedure
3           ' Add input parameter and supply a value for it
4           command.Parameters.Add("@TypeOfBooksID", SqlDbType.Int, 4)
5           command.Parameters("@TypeOfBooksID").Value = TypeOfBooksID
6           Try
7               ' Open the connection
8               connection.Open()
9               ' Return an SqlDataReader to the calling function
0               Return command.ExecuteReader(CommandBehavior.CloseConnection)
1           Catch ex As Exception
2               ' Close the connection and throw exception
3               connection.Close()
4               Throw ex
5           End Try
6       End Function
```

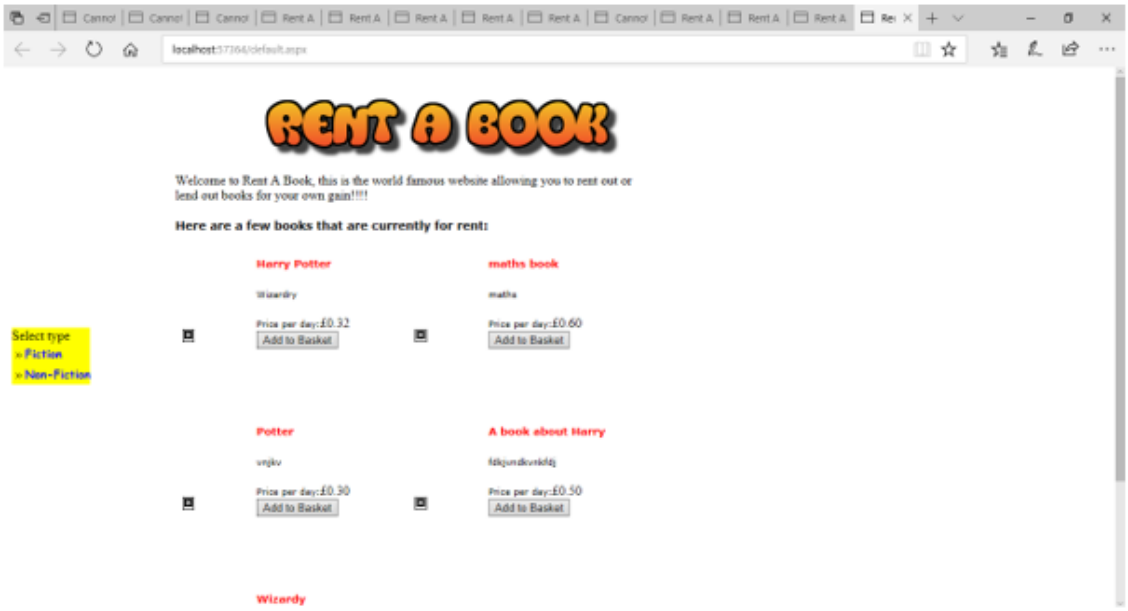
selected that have been selected to be on the type of boo promotion, name, description and

price per day. The stored procedure also takes in the parameter TypeOfBookID. The function will then return it, where it will be returned to the list and displayed to the user.



This function executes the procedure GetBooksOnGenrePromotion which will return all of the books in the table TheBooks, that have a 1 in the column OnGenreOfBookPromotion. The function will then recive this data and returns it, where it is saved to list and displayed to the user.

```
Public Shared Function GetBooksOnGenreOfBookPromotion() As SqlDataReader
    ' Create the connection object
    Dim connection As New SqlConnection(connectionString)
    ' Create and initialize the command object
    Dim command As New SqlCommand("GetBooksOnGenrePromotion", connection)
    command.CommandType = CommandType.StoredProcedure
    Try
        ' Open the connection
        connection.Open()
        ' Return an SqlDataReader to the calling function
        Return command.ExecuteReader(CommandBehavior.CloseConnection)
    Catch ex As Exception
        ' Close the connection and throw exception
        connection.Close()
        Throw ex
    End Try
End Function
```



Test Results / Evidence

What is being tested	***Input***	***Justification of input***	***Expected outcome***	***Actual outcome***	***How to solve***	***Proof***
The Database integrity	Setting genreID to invalid TypeofBookID	If an invalid type of book ID is entered the program must recognise this and not allow it to be saved	Error	Error	N/A	Image 1
Does home page load	N/A	N/A	Loaded homepage	Loaded homepage	N/A	Image 2
Does list of genres appear when type of book is selected	Fiction is selected	Fiction is one of the type of books meaning the list of genres should appear when clicked	List of genres to appear	List of genres appeared	N/A	Image 3
Do the books appear when a genre is selected	Action and adventure is selected	Action and adventure is one of the genres belonging to fiction	List of action and adventure books to appear	Image 4	I needed to change the parameters name in the SQL stored procedure to GenreOfBookID	Image 5

Image 1:

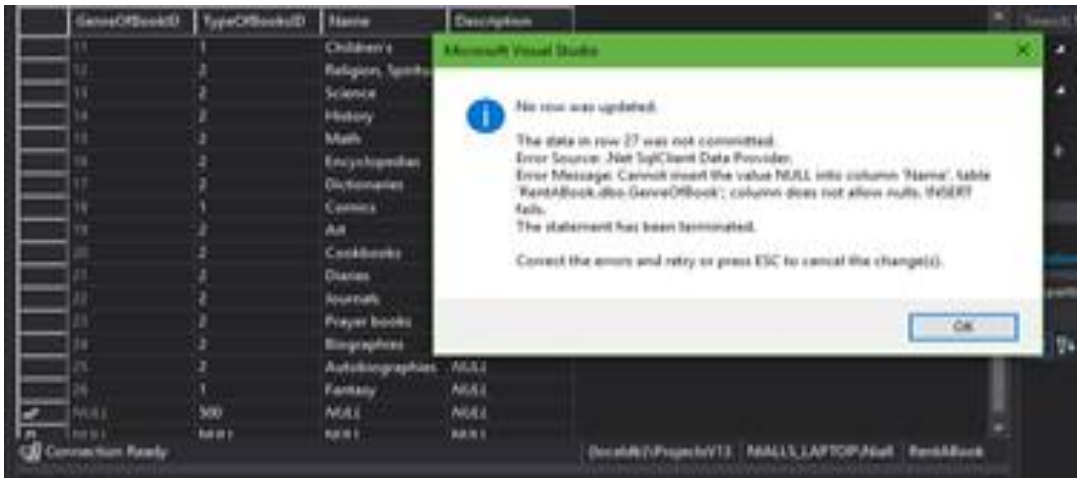


Image 2:

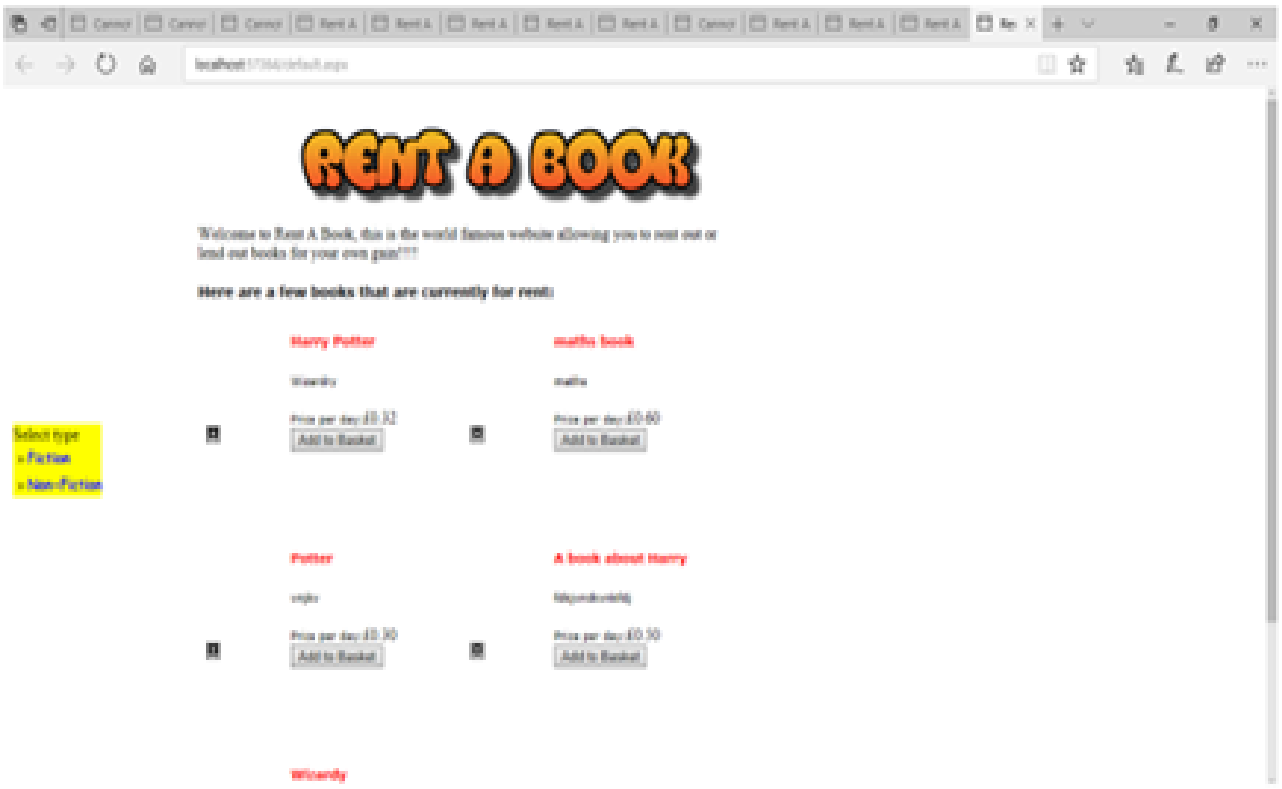


Image 3:

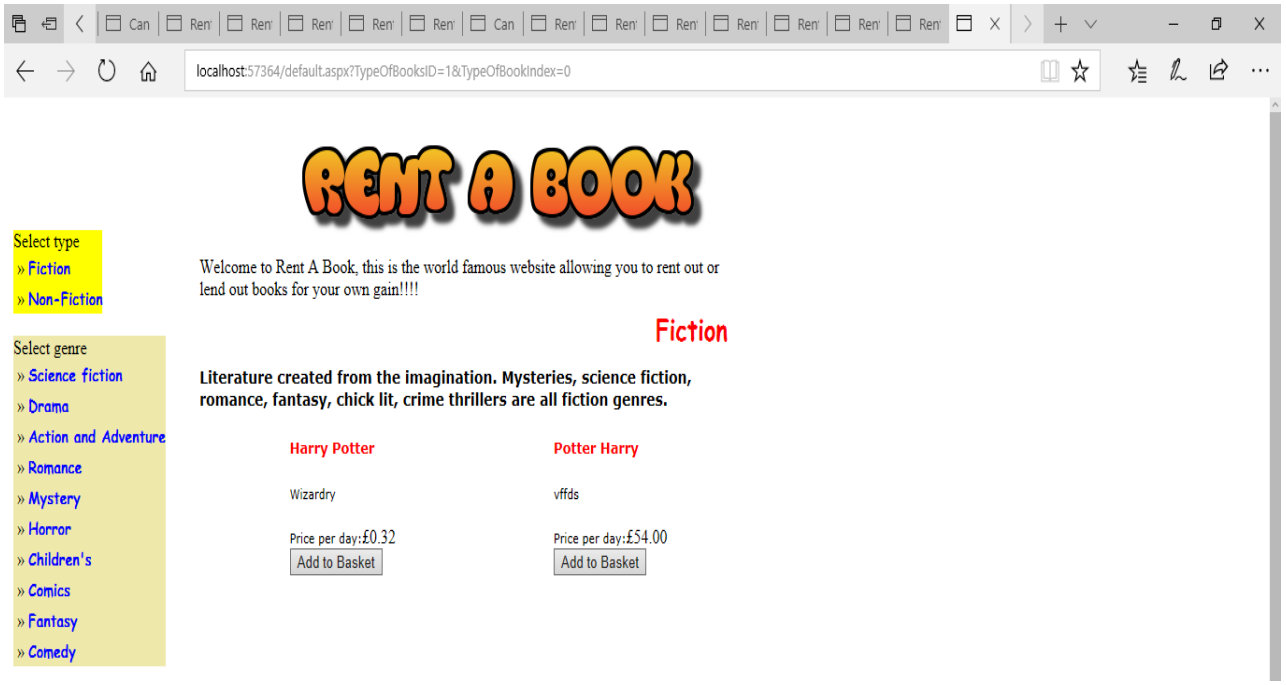


Image 4:

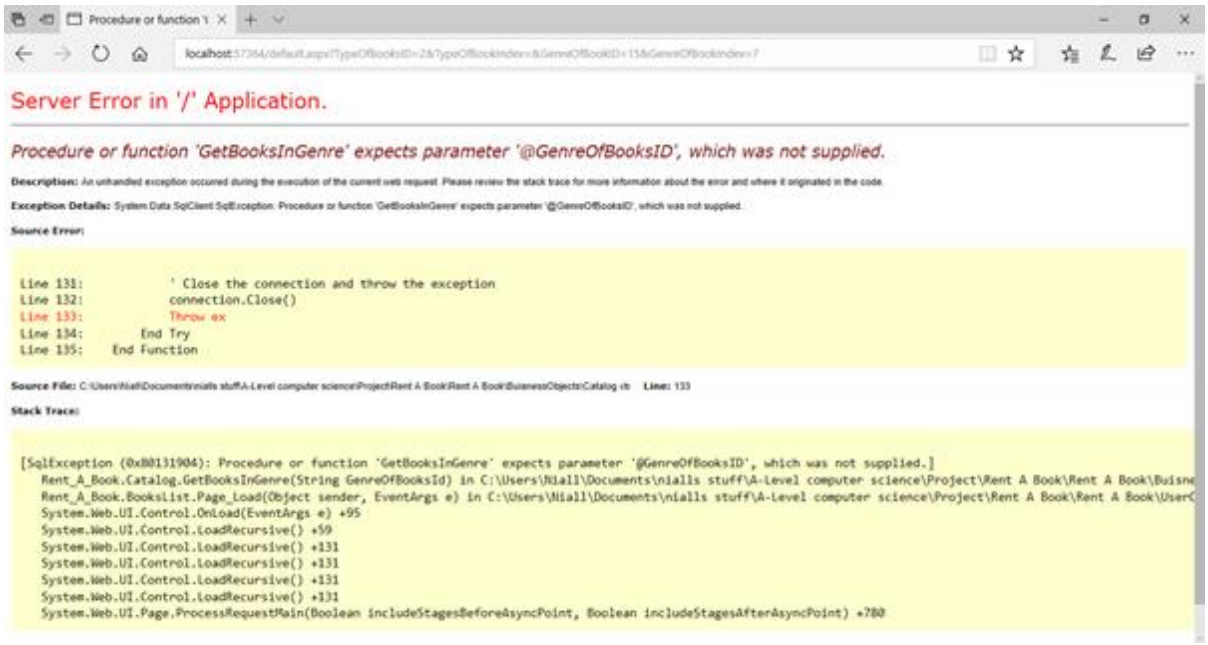
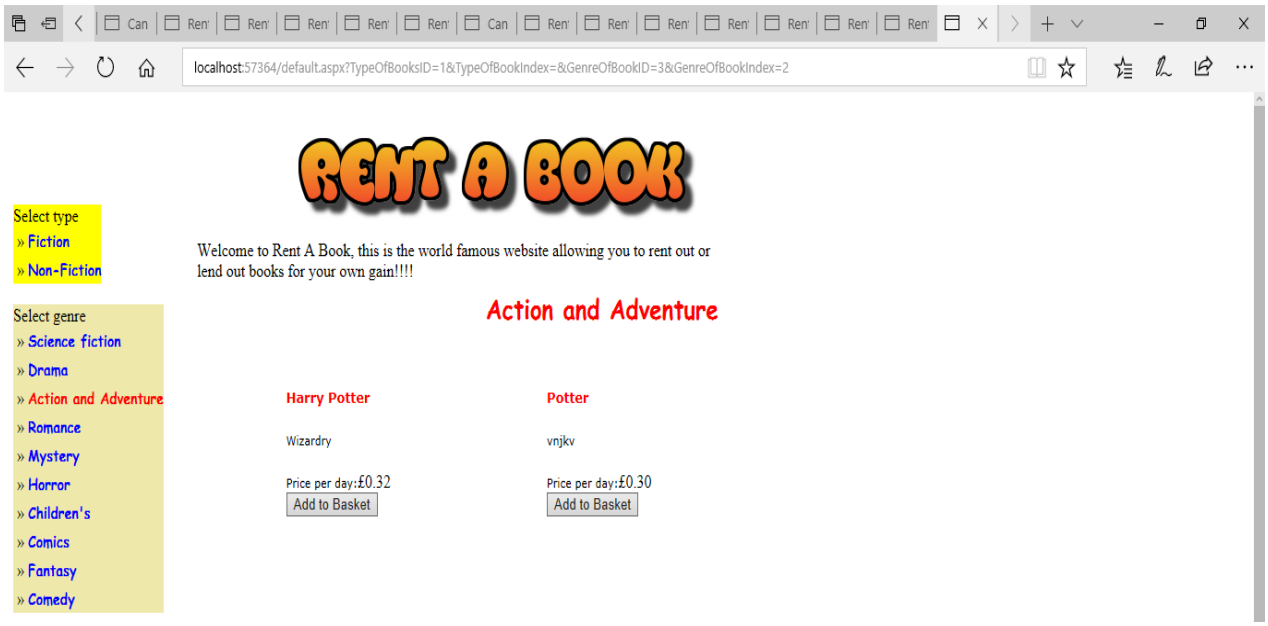


Image 5:



Review

This prototype was an overall success as it met every aim stated at the beginning of the iteration. It is able to load the webpage and is able to display the books selected to be viewed on the main homepage. I was also successfully able to create the one to many relationship between the books and genres, this has allowed the user to be able to click on a specific genre and view the correct books associated to the genre. The SQL procedures I wrote also worked correctly as they were able to query and give to the user the correct type of books they wanted. The layout of the webpage has also enabled the user to navigate to the genre of book they want easily without much hassle. The tests I preformed allowed me to find and fix errors both logically and syntax.

Stakeholder feedback

User Interface

I like the front page and especially the company logo very clearly displayed at the top of the page. It provides all the main functionality a user will normally require when visiting the site. This means that the user can very quickly search for books they may want to rent by navigating the menu. I also like the idea of recommending a few books to users who visit the site. This may provide an income stream by advertising.

I like that the pages are kept uncluttered and clear for users to easily navigate. I would however like to see a text search facility on the front page. There should also be a link to add new books and register new users.

Functionality

The book menus work well and all the links are working as expected. I want a text search facility that will allow users to not only search book titles, but also authors, descriptions, genres etc.

The site also needs a sign-up page to register new users. This page should store users' email addresses so that we can contact them when a rental has been matched, or to advertise. If possible, a watchlist would be great whereby users can add books they may want to look at later.

Iteration 2 - Date 25/11/17

Aims for this iteration

In this prototype, I aim to allow the user to search for specific books, by implementing a search bar into my website. The search bar will need not only to search for the word the user searches for in the book titles, but also in their descriptions. The results must then come back in ranking order, the first book being closest to what was searched for. The prototype must also allow the users to add books to their basket. The basket must also allow the option for the user to change the number of days they want to rent for and remove the book from the basket if the user chooses to do so.

Functionality that the prototype will have

In order for this prototype to work I will need to have the program create a temporary table for it to store all of the books that relate to what was searched for. Also it will need to be able to split up a sentence as most users search a string of words instead of just one. In order to save each users basket I must make the program use the user's cookies. The basket and search bar will also require separate user controls.

Annotated code screenshots with description

Making the SQL stored procedure

To make the search bar for my website I only needed one stored procedure, the SearchCatalog procedure. This procedure will return the number of results found. It will first create a temporary

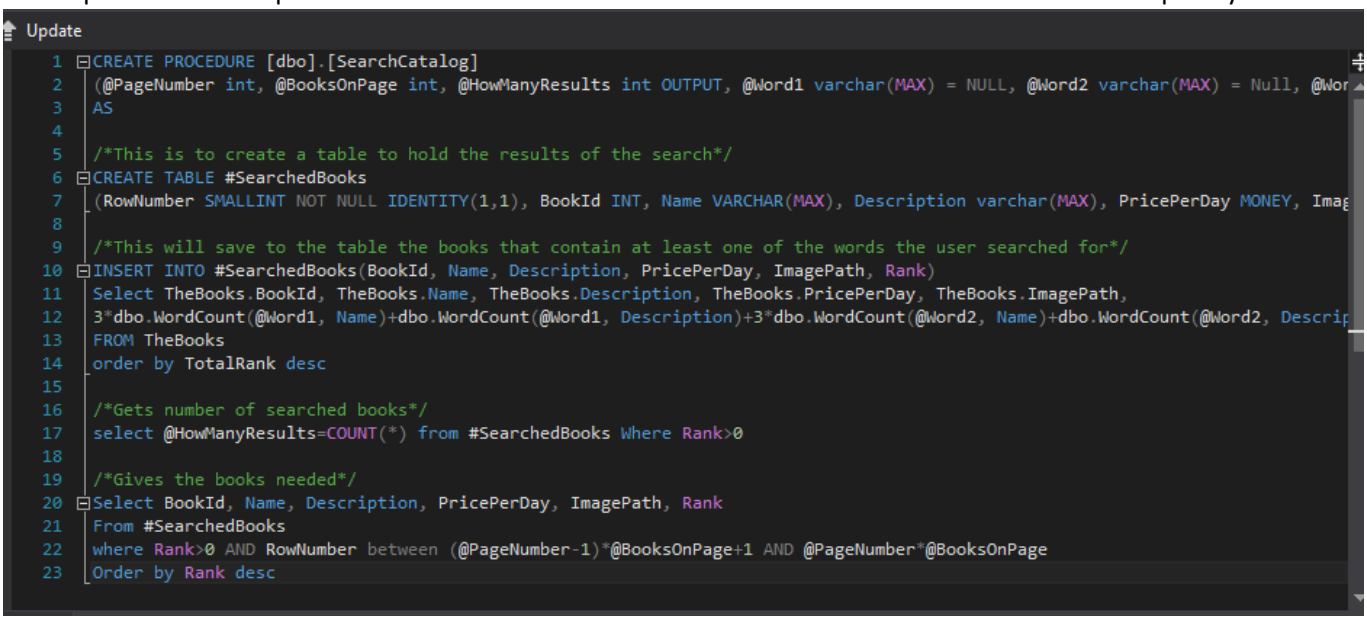
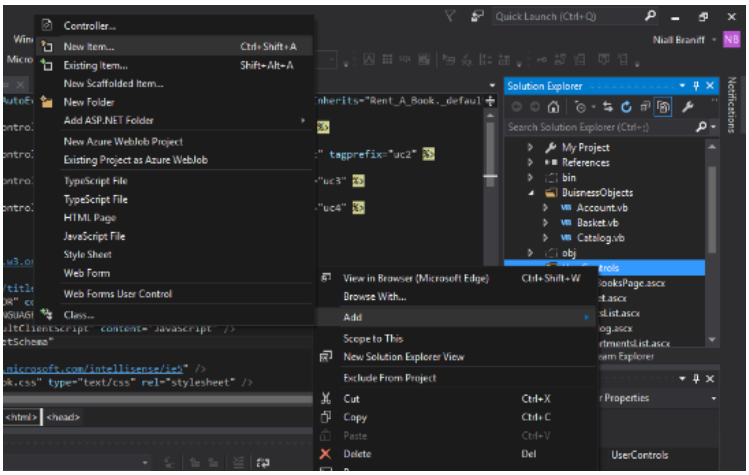
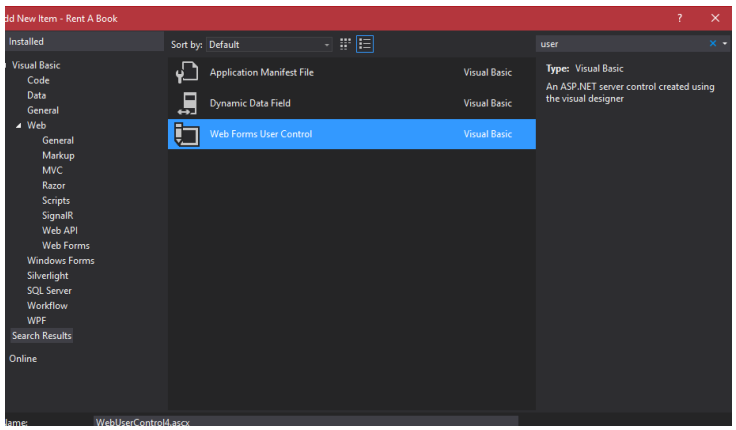


table called #SearchedBooks. It needs a # in front of it to say it is a temporary table. The table will have five columns, RowNumber, BookId, Name, Description, PricePerDay and ImagePath. The procedure will then insert into the temporary table all of the books/ records that have at least one of the words the user searched for. It will also rank them in order depending on how many times the word was found in the name or description with name being a priority one. The stored procedure will then return the books needed.

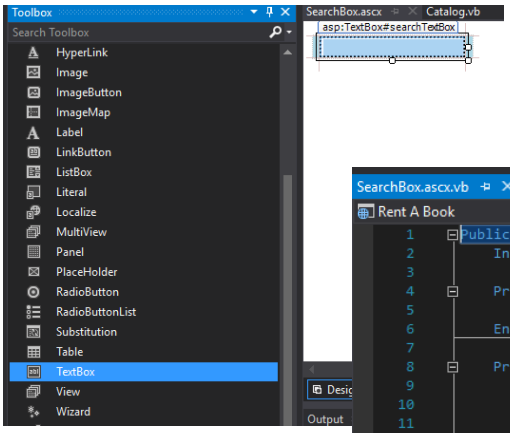
Displaying the search bar and the search results

To display the search bar, I first needed to make a user control. To do this I right clicked on solution explorer, clicked on add and then selected Web Forms User Control and named it SearchBox.ascx.



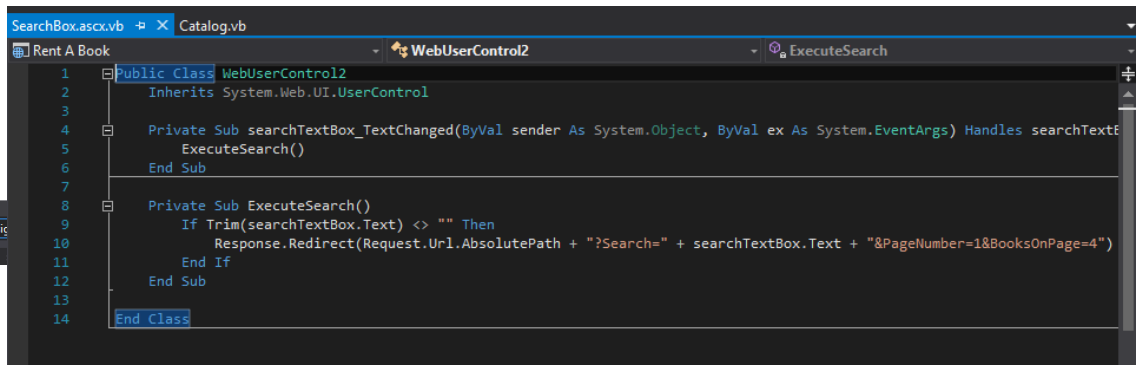


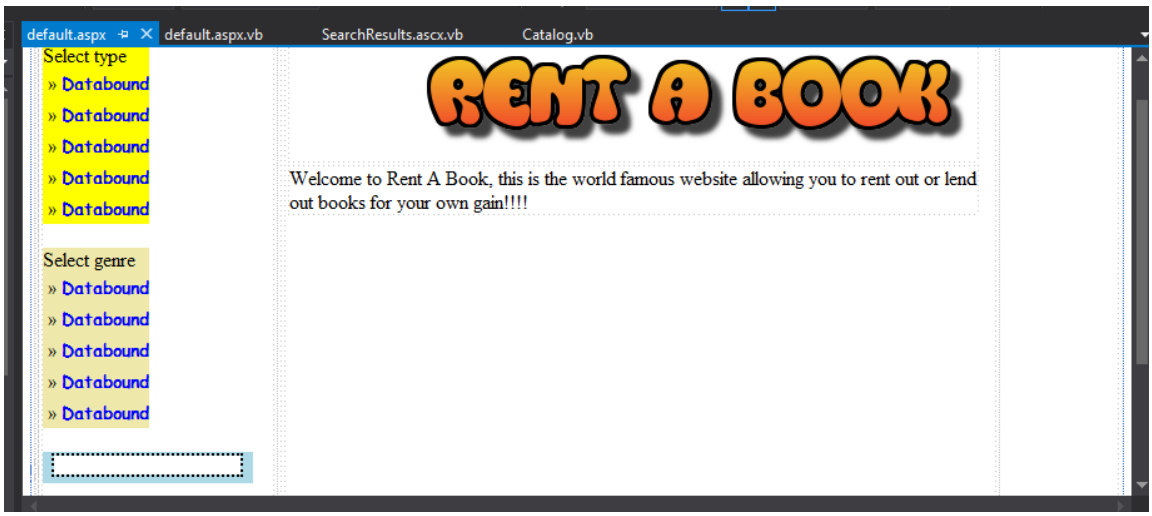
I then went onto design view and dragged and dropped a textbox from the toolbox onto the page. Next, I named its ID as SearchBox.



After that I then accessed the code behind file and edited it to this.

The code written will first check to see if the text in the text box has been changed. If it has it will then run ExecuteSearch(). This will then redirect the page URL. I then dragged and dropped the file onto default.aspx in design view.



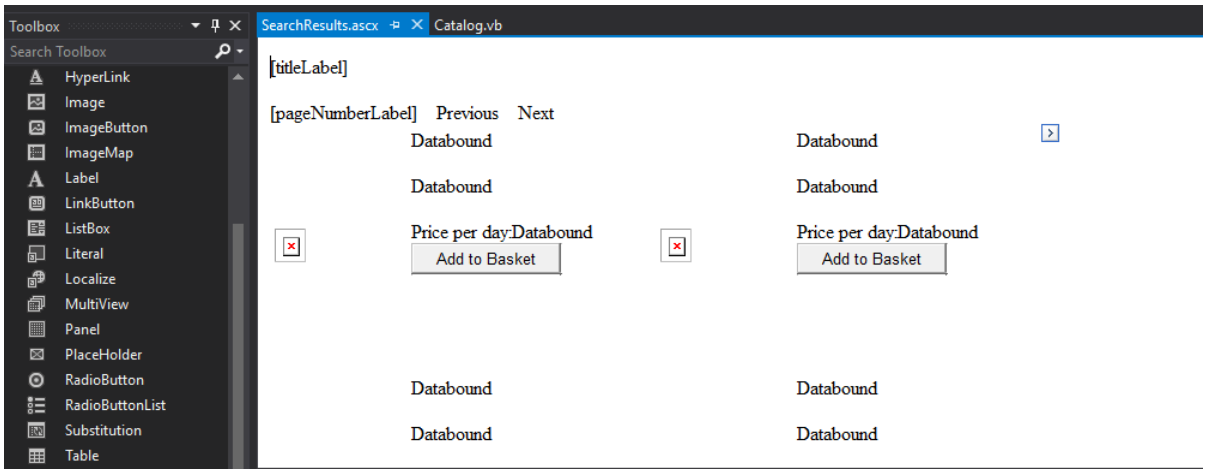


After that I needed to go onto the code behind file of default.aspx, by right clicking it in solution explorer and selecting view code. I then added another if statement checking to see if the user has searched for something. If they had it will then load SearchResults.ascx into the page contents cell.

```
Public Class _default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
        Dim searchString As String = Request.QueryString("Search")
        If Not searchString Is Nothing Then
            Dim control As Control
            control = Page.LoadControl("UserControls/SearchResults.ascx")
            pageContentsCell.Controls.Add(control)
        ElseIf Not TypeOfBooksID Is Nothing Then
            Dim control As Control
            control = Page.LoadControl("UserControls/Catalog.ascx")
            pageContentsCell.Controls.Add(control)
        Else
            Dim control As Control
            control = Page.LoadControl("UserControls/FirstPage.ascx")
            pageContentsCell.Controls.Add(control)
        End If
    End Sub
```

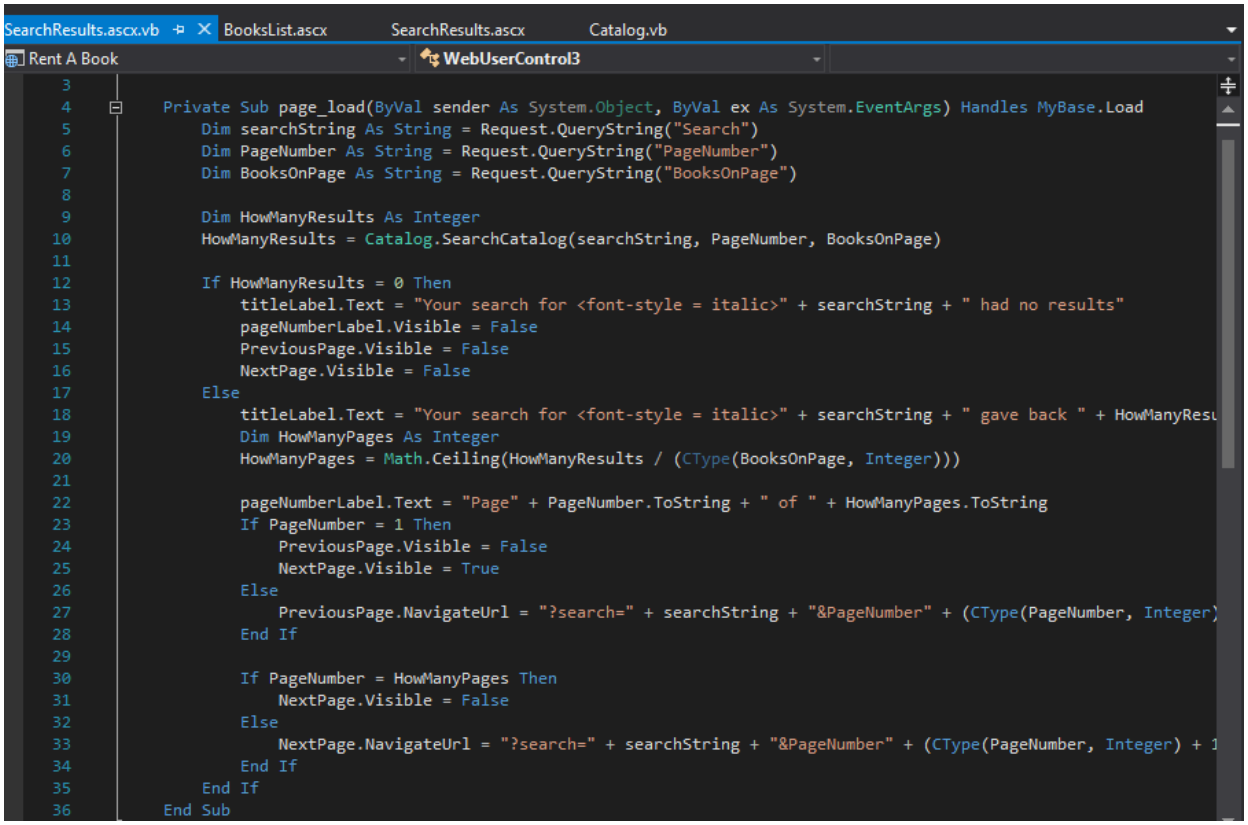
By doing this meant that I now needed to make another user control called SearchResults.ascx. so, I right clicked on solution explorer, clicked on add and selected web user control form, then naming it SearchResults.ascx.

I then went onto design view and added two labels (setting one ID to titleLable and the other to

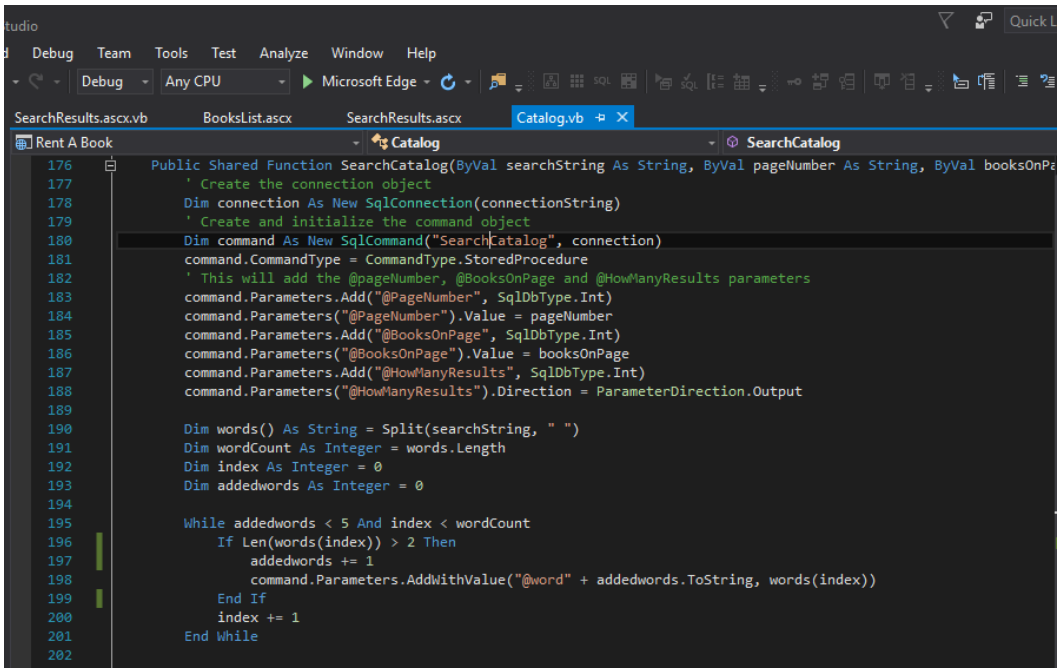


pageNumberLabel), two hyperlinks (one called Previous and the other called Next). I then dragged and dropped the BooksList.ascx file onto SearchResults.ascx.

After that I went onto the code behind file of SearchResults.ascx and edited it as follows. The code will first set the variables searchString (containing the users search), PageNumber (the page number



the user is on) and BooksOnPage (the number of books on any one page). It will then call the function SearchCatalog from the Catalog class to find out how many results there are. If there are no results then it will hide the two hyperlinks and the pageNumberLabel and set the titleLabel to inform the user that there were no results regarding their search. However, if there are results then it will work out how many pages are needed, set the titleLabel to tell the user how many results were found and display the next and previous page option if needed.



```
203         Try
204             connection.Open()
205             Dim reader As SqlDataReader
206             reader = command.ExecuteReader(CommandBehavior.CloseConnection)
207             Dim table As New DataTable()
208             Dim fieldCount As Integer = reader.FieldCount
209             Dim fieldindex As Integer
210             For fieldindex = 0 To fieldCount - 1
211                 table.Columns.Add(reader.GetName(fieldindex), reader.GetFieldType(fieldindex))
212             Next
213             Dim row As DataRow
214             While reader.Read()
215                 row = table.NewRow()
216                 For fieldindex = 0 To fieldCount - 1
217                     row(fieldindex) = reader(fieldindex)
218                 Next
219                 table.Rows.Add(row)
220             End While
221             reader.Close()
222             HttpContext.Current.Session("searchTable") = table
223             Return command.Parameters("@HowManyResults").Value
224         Catch ex As Exception
225             connection.Close()
226             Throw ex
227         End Try
228     End Function
229 End Class
```

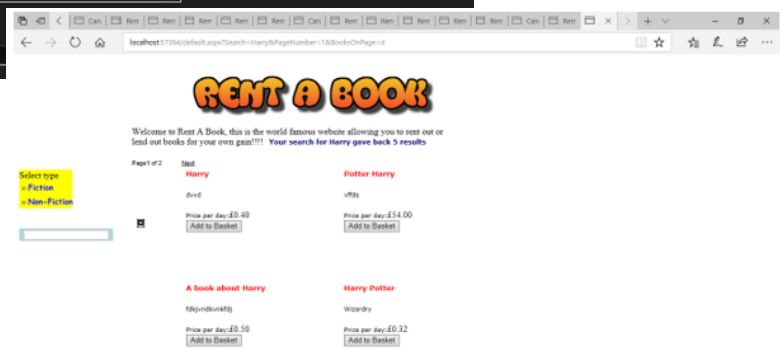
Afterwords I made a new function called SearchCatalog in the class Category. This function is used to execute the SQL stored procedure called SearchCatalog. It also sets the parameters to the page number, the ammount of books on a page and how many results. It will then split up the search string and send each individual word through as a parameter to the procedure. It will then save the search results and column information from the Sql dataReader to a Data table. Afterwords it will close the reader and save the results to the current session and return the number of books found.

Finaly I needed to update the BookList.ascx file so that it will load the results found from the users search.

SearchResults.ascx.vb | BooksList.ascx | SearchResults.ascx | Catalog.vb | BooksLis

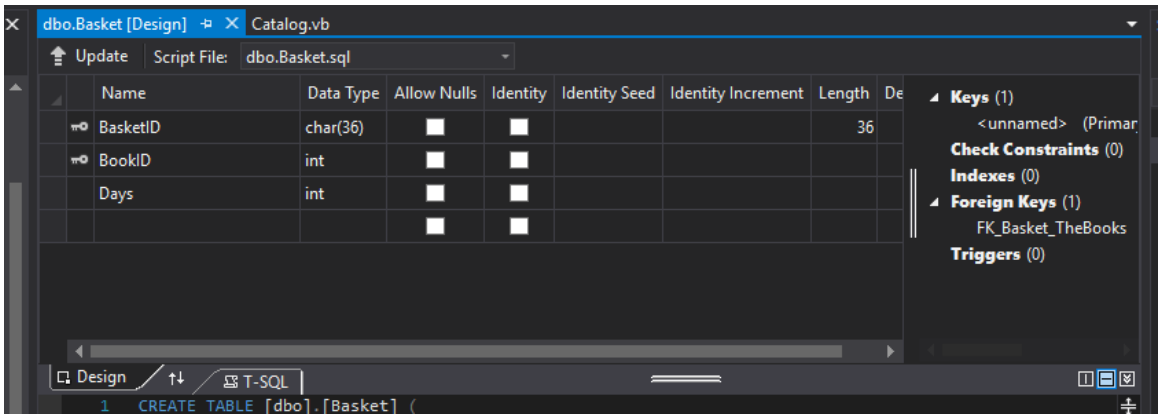
Rent A Book | (BooksList Events) | Load

```
1   Public Class BooksList
2       Inherits System.Web.UI.UserControl
3
4       Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
5           Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
6           Dim GenreOfBookID As String = Request.QueryString("GenreOfBookID")
7           Dim searchString As String = Request.QueryString("Search")
8
9           If Not searchString Is Nothing Then
10              List.DataSource = Session("searchTable")
11              List.DataBind()
12              Session.Remove("searchTable")
13          ElseIf Not GenreOfBookID Is Nothing Then
14              List.DataSource = Catalog.GetBooksInGenre(GenreOfBookID)
15              List.DataBind()
16          ElseIf Not TypeOfBooksID Is Nothing Then
17              List.DataSource = Catalog.GetBooksOnTypeOfBookPromotion(TypeOfBooksID)
18              List.DataBind()
19          Else
20              List.DataSource = Catalog.GetBooksOnGenreOfBookPromotion()
21              List.DataBind()
22          End If
23      End Sub
```



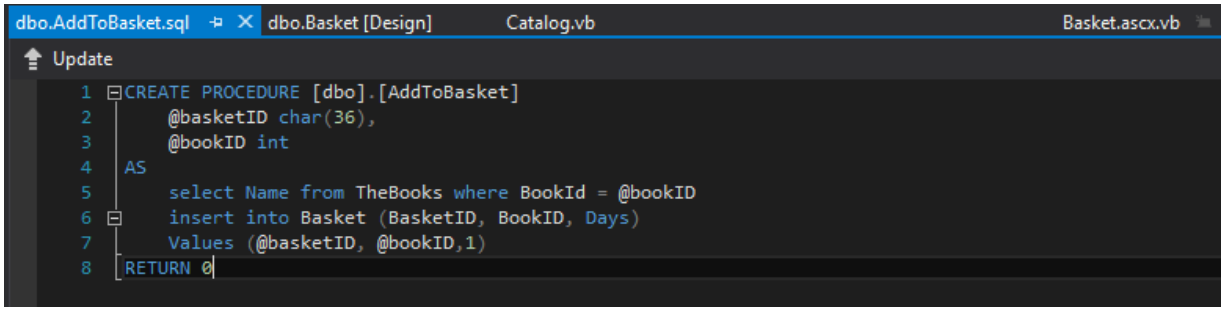
Making the users basket

To make the basket I first made a table called Basket in the RentABook database. The table will hold the BasketID, the BookID and the number of Days rented for. To do this I first right clicked on tables in the RentABook database and then I clicked on Add New Table. Afterword's I added the columns,



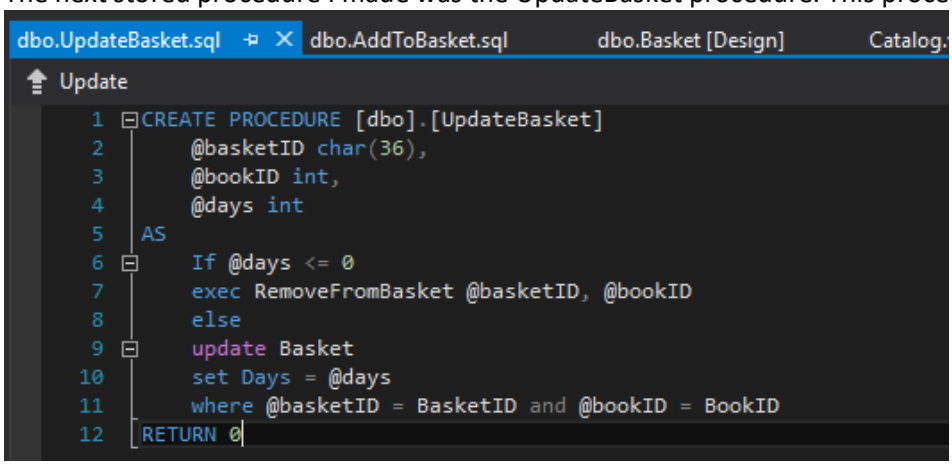
BasketID, BookID and Days. I set the BookID column to be the foreign key so that there will be a relationship between the Basket and the TheBooks.

After making the table I then made the three SQL stored procedures needed, the AddToBasket procedure, the UpdateBasket procedure and the RemoveFromBasket procedure. To make the AddToBasket procedure I needed to right click on stored procedures and then I clicked on Add New Stored Procedure. Next I edited the procedure as follows. The procedure will take in two parameters, one which is the @basketID and the other is the @bookID. The procedure will then



select the name of the book in TheBooks table where BookId equals @bookID. After that it will insert @basketID, @bookID and 1 in to the table called Basket under the columns BasketID, BookID andDays respectively.

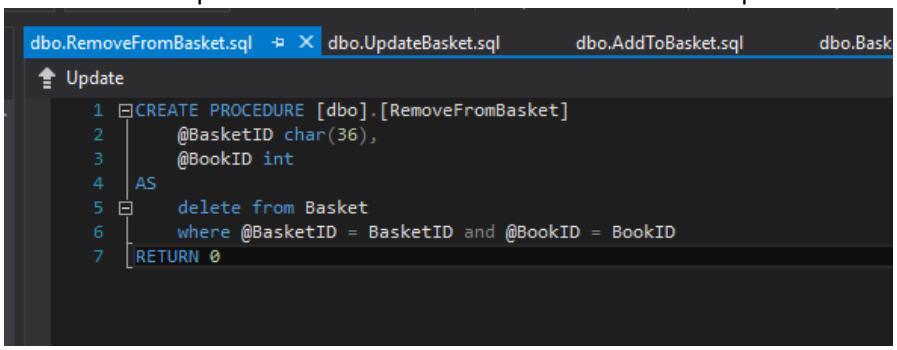
The next stored procedure I made was the UpdateBasket procedure. This procedure will take in the



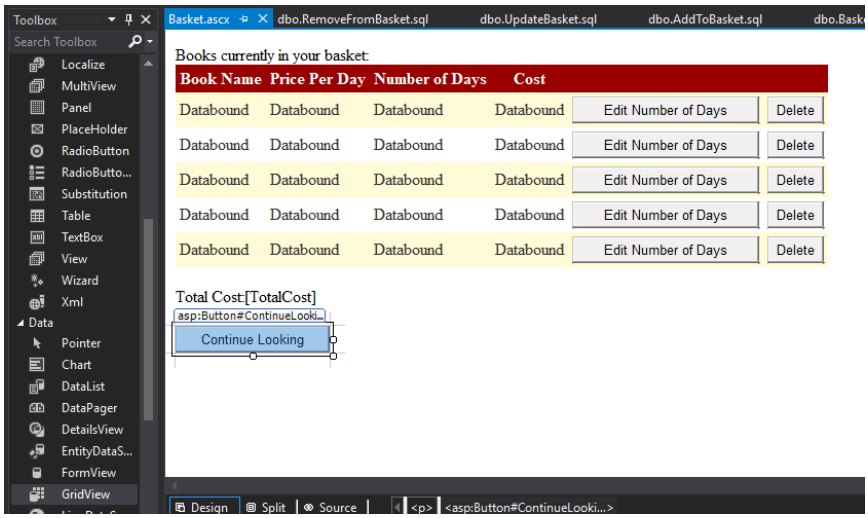
basket's ID, The Book's ID and the number of days as its parameters. It wil lthen check if the user has selected 0 or less than 0 days for the book. If it has it will execute the stored procedure called

RemoveFromBasket, passing in the parameters called @basketID and @bookID. Otherwise it will set the number of days to the ammount of days the user has requested.

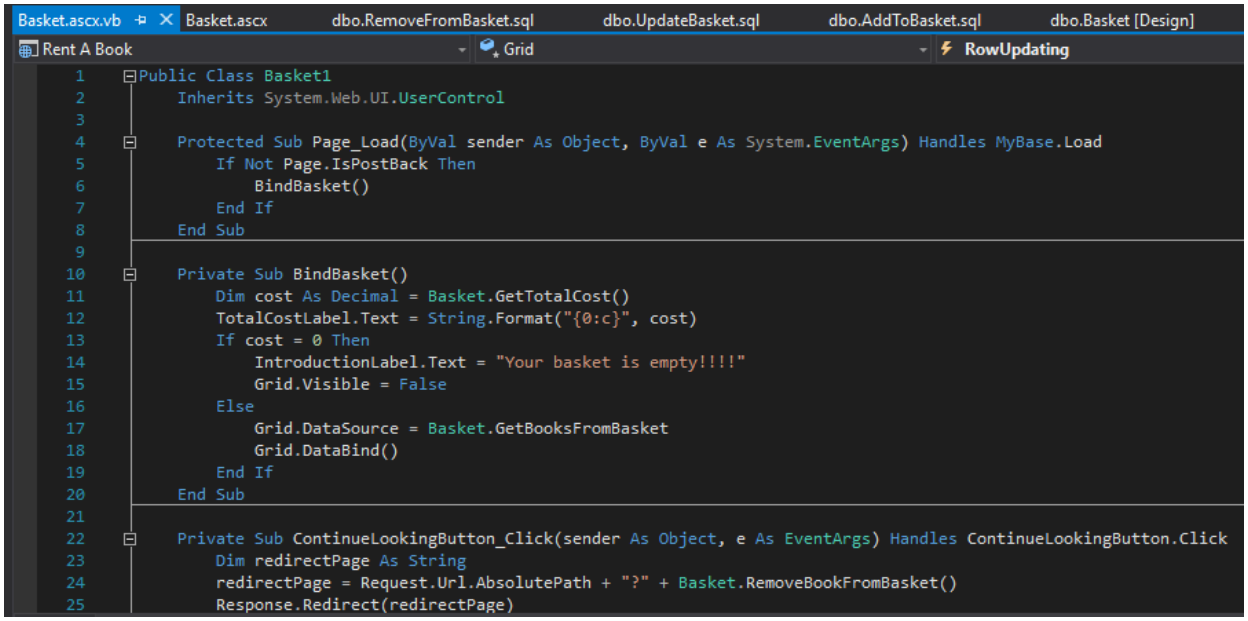
The final stored procedure nedded is the RemoveFromBasket procedure. This procedure takes in the parameters BasketID and the parameter BookID. It then deletes where the @bookID equals BookID and @basketID equals BasketID from the Basket table.



After making the stored procedures, I then went onto making the necessary user control called basket.ascx. I then went onto design view and dropped grid view from the toolbox along with the three labels and the continue looking button.



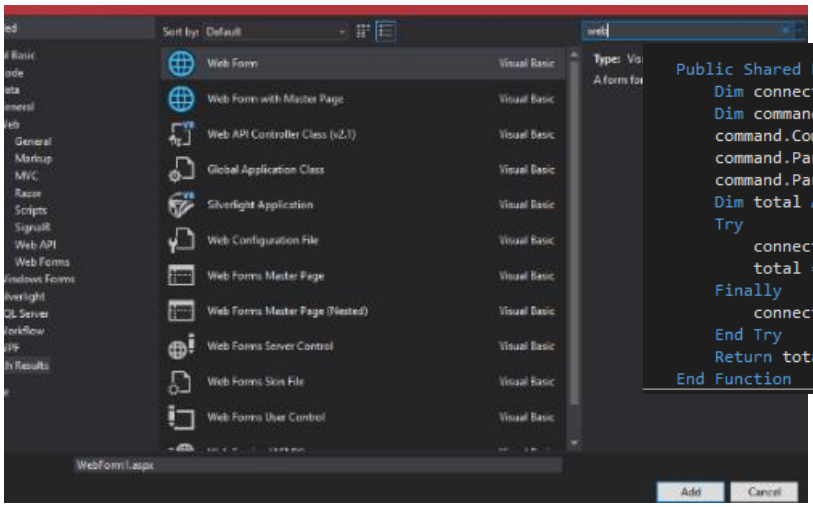
I then accessed the code behind file of the basket.ascx and added seven pieces of code to it. The first one I added was the Page_load one where it checks to see if it the page has been rendered for the first time or if it is the result of a postback. If it is being rendered for the first time then it will run



```
    End Sub

    Private Sub Grid_RowCancelingEdit(sender As Object, e As GridViewCancelEditEventArgs) Handles Grid.RowCancelingEdit
        Grid.EditIndex = -1
        BindBasket()
    End Sub

    Private Sub Grid_RowDeleting(sender As Object, e As GridViewDeleteEventArgs) Handles Grid.RowDeleting
        Dim row As GridViewRow = Grid.Rows(e.RowIndex)
        Dim bookId As String = Grid.DataKeys(e.RowIndex).Value
        Basket.RemoveFromBasket(bookId)
        BindBasket()
    End Sub

    Private Sub Grid_RowEditing(sender As Object, e As GridViewEditEventArgs) Handles Grid.RowEditing
        Grid.EditIndex = e.NewEditIndex
    End Sub

    Private Sub Grid_RowUpdating(sender As Object, e As GridViewUpdateEventArgs) Handles Grid.RowUpdating

        Dim row As GridViewRow = Grid.Rows(e.RowIndex)

        Dim bookId As String = Grid.DataKeys(e.RowIndex).Value

        Dim days As String = TryCast(row.FindControl("TextBox1"), TextBox).Text

        Try
            Basket.UpdateBasket(bookId, days)
        Catch ex As Exception
            IntroductionLabel.Text = "You can only enter valid number of days"
        Finally
            Grid.EditIndex = -1
            BindBasket()
        End Try
    End Sub
```

BindBasket(). BindBasket() will work out the cost from the GetTotalCost function from the Basket class. It will then check to see if cost equals 0, if it does then it will tell the user that their basket is empty, otherwise it will call the function GetBooksFromBasket from the Basket class. I then wrote the behind code for when the user wishes to cancel editing a book by setting the grids EditIndex to minus one. When deleting a row, the code will call the function RemoveFromBasket with a parameter called bookId. If the user wants to update the row, it will first take in the number of days the user has requested and then calls the UpdateBasket function with the parameters bookId and days. If the user tries to input an invalid number of days the code will catch it and then tells the user that “You can only enter valid number of days.”

After this I then made Basket.vb file by right clicking and adding a Web form and calling it Basket.vb. the first function I made in the class Basket was the GetTotalCost function. This function executes the SQL stored procedure called Total cost with the parameter basket ID. The function will the return the results of this.




```
Public Shared Function GetTotalCost() As Decimal
    Dim connection As New SqlConnection(connectionString)
    Dim command As SqlCommand = New SqlCommand("TotalCost", connection)
    command.CommandType = CommandType.StoredProcedure
    command.Parameters.Add("@basketID", SqlDbType.Char, 36)
    command.Parameters("@basketID").Value = basketId
    Dim total As Decimal = 0
    Try
        connection.Open()
        total = command.ExecuteScalar()
    Finally
        connection.Close()
    End Try
    Return total
End Function
```

After that I made the GetBooksFromBasket function which executes the GetBooksForBasket procedure which takes in the basket's ID as its parameter. The next function I made after this was the RemoveFromBasket one. This function calls the RemoveFromBasket SQL stored procedure that has a parameter called @basketID. The final function I made in the class Basket, was the UpdateBasket function. This function opens the connection to the stored SQL procedure called UpdateBasket. This procedure takes in the basketID, the bookID and the number of days in as parameters.

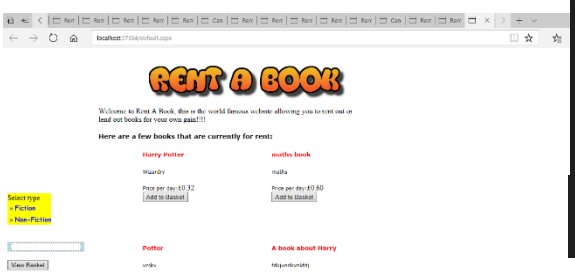
```
Public Shared Function GetBooksFromBasket() As SqlDataReader
    Dim connection As New SqlConnection(connectionString)
    Dim command As New SqlCommand("GetBooksForBasket", connection)
    command.CommandType = CommandType.StoredProcedure
    command.Parameters.Add("@basketID", SqlDbType.Char, 36)
    command.Parameters("@basketID").Value = basketId
    Try
        connection.Open()
        Return command.ExecuteReader(CommandBehavior.CloseConnection)
    Catch ex As Exception
        connection.Close()
        Throw ex
    End Try
End Function
```

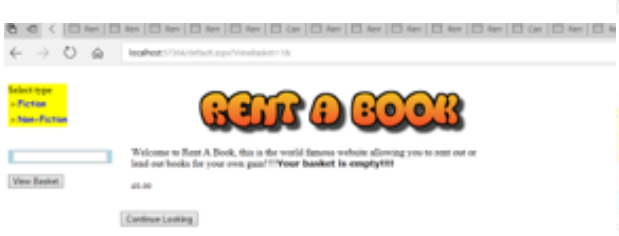
```
Public Shared Function RemoveFromBasket(ByVal bookId As String)
    Dim connection As New SqlConnection(connectionString)
    Dim command As New SqlCommand("RemoveFromBasket", connection)
    command.CommandType = CommandType.StoredProcedure
    command.Parameters.Add("@basketID", SqlDbType.Char, 36)
    command.Parameters("@basketID").Value = basketId
    command.Parameters.Add("@BookID", SqlDbType.Int)
    command.Parameters("@BookID").Value = bookId
    Try
        connection.Open()
        command.ExecuteNonQuery()
    Finally
        connection.Close()
    End Try
    Return 0
End Function
```

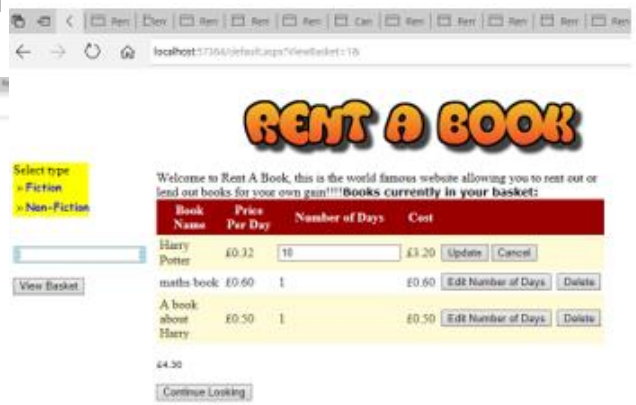
Finaly the last part of this was too update the default.aspx behind file by making it check if the view basket button has been clicked so that it knows when to add the user contol basket.ascx to the pageContentsCell.

```
Public Shared Function UpdateBasket(ByVal bookId As String, ByVal days As Integer)
    Dim connection As New SqlConnection(connectionString)
    Dim command As New SqlCommand("UpdateBasket", connection)
    command.CommandType = CommandType.StoredProcedure
    command.Parameters.Add("@basketID", SqlDbType.Char, 36)
    command.Parameters("@basketID").Value = basketId
    command.Parameters.Add("@BookID", SqlDbType.Int)
    command.Parameters("@BookID").Value = bookId
    command.Parameters.Add("@Days", SqlDbType.Int)
    command.Parameters("@Days").Value = days
    Try
        connection.Open()
        command.ExecuteNonQuery()
    Finally
        connection.Close()
    End Try
    Return 0
End Function
```

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
    Dim searchString As String = Request.QueryString("Search")
    Dim ViewBasket As String = Request.QueryString("ViewBasket")
    If Not ViewBasket Is Nothing Then
        Dim control As Control
        control = Page.LoadControl("UserControls/Basket.ascx")
        pageContentsCell.Controls.Add(control)
    ElseIf Not searchString Is Nothing Then
        Dim control As Control
        control = Page.LoadControl("UserControls/SearchResults.ascx")
        pageContentsCell.Controls.Add(control)
    ElseIf Not TypeOfBooksID Is Nothing Then
        Dim control As Control
        control = Page.LoadControl("UserControls/Catalog.ascx")
        pageContentsCell.Controls.Add(control)
    Else
        Dim control As Control
        control = Page.LoadControl("UserControls/FirstPage.ascx")
        pageContentsCell.Controls.Add(control)
    End If
End Sub
Protected Sub ViewBasketButton_Click(sender As Object, e As EventArgs) Handles ViewBasketButton.Click
    If Request.QueryString("ViewBasket") Is Nothing Then
        Response.Redirect("default.aspx?ViewBasket=1&" & Request.QueryString.ToString())
    End If
End Sub
```







Test Results / Evidence

What is being tested	***Input***	***Justification of input***	***Expected outcome***	***Actual outcome***	***How to solve***	***Proof***
Search bar	Harry Potter	There is a book called Harry Potter that already exists in the database	List of Harry Potter books	Image 1	The program was looking in whatever database it's currently connected to, not the temp table, to fix this I needed to put # in front of the temp tables name	Image 2
View basket	View basket clicked	Clicking on the view basket button is the only way of viewing the basket	Basket to appear	The basket appeared	N/A	Image 3
Add to basket	Add to basket clicked	Clicking on the add to basket button is the only way of adding a book	The book to appear in the basket	The book appeared in the basket	N/A	Image 4 to 5
Update basket	Number of days	The number of days that the user wants to rent for	Number of days to change	The number of days changed	N/A	Image 6 to 7
Delete from basket	Clicking the delete button	The delete button is the only way to remove a book from the basket	The book to be removed from basket	The book was removed from the basket	N/A	Image 5 and 8

Image 1:

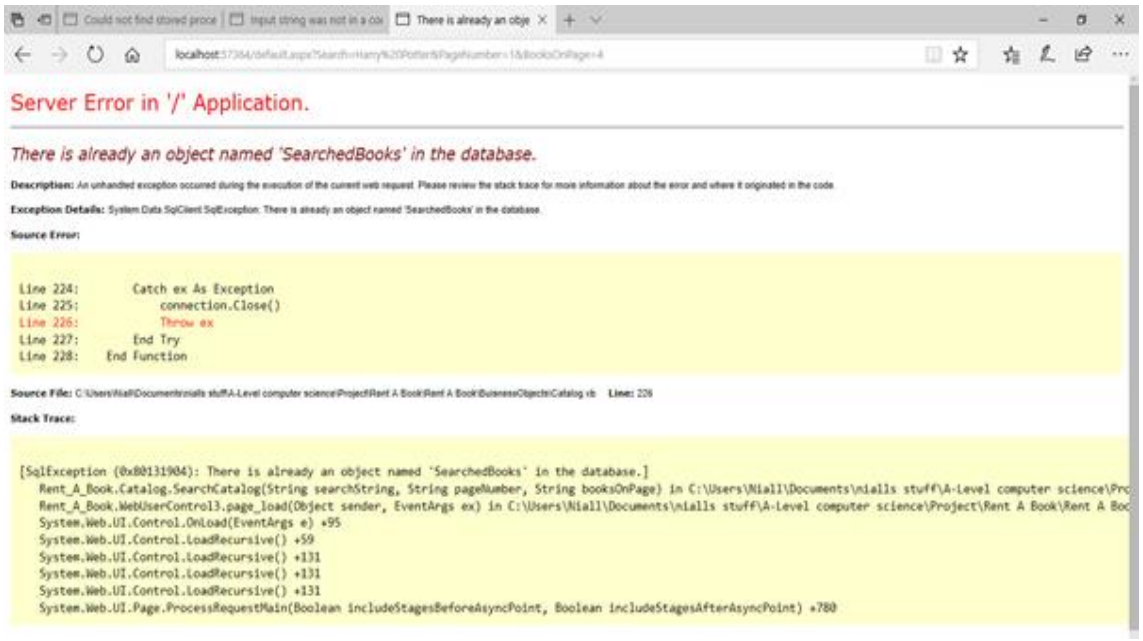


Image 2:

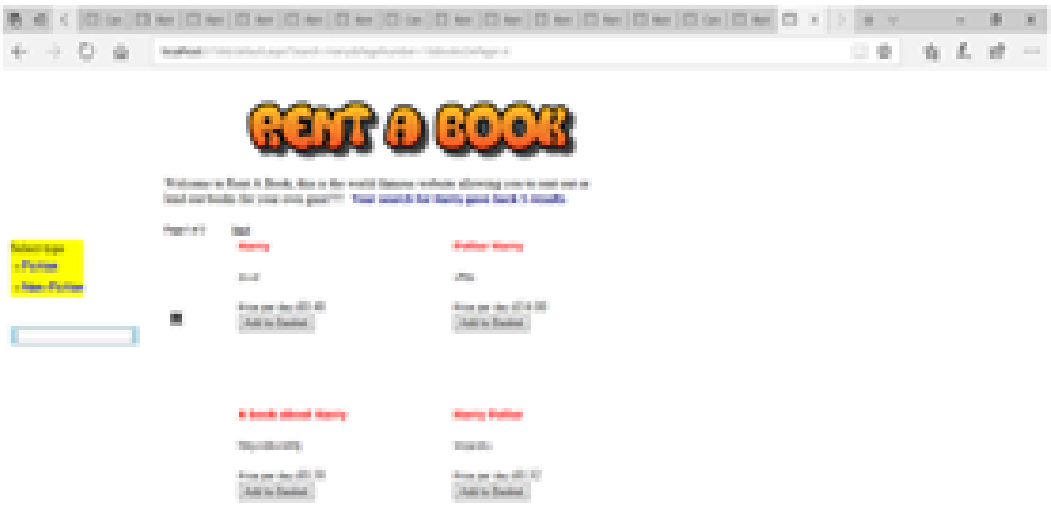


Image 3:

Image 4:

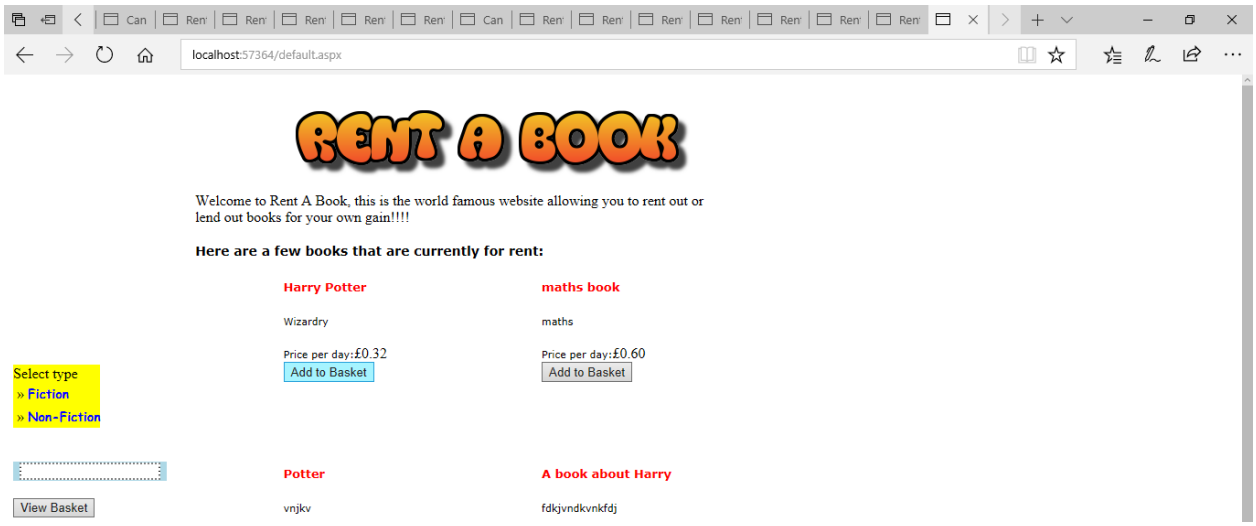


Image 5:

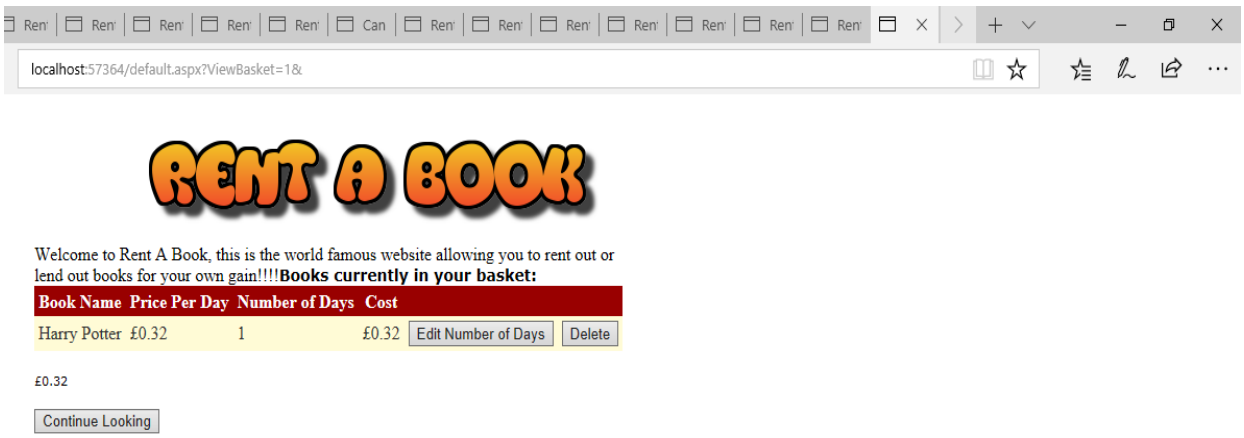


Image 6:

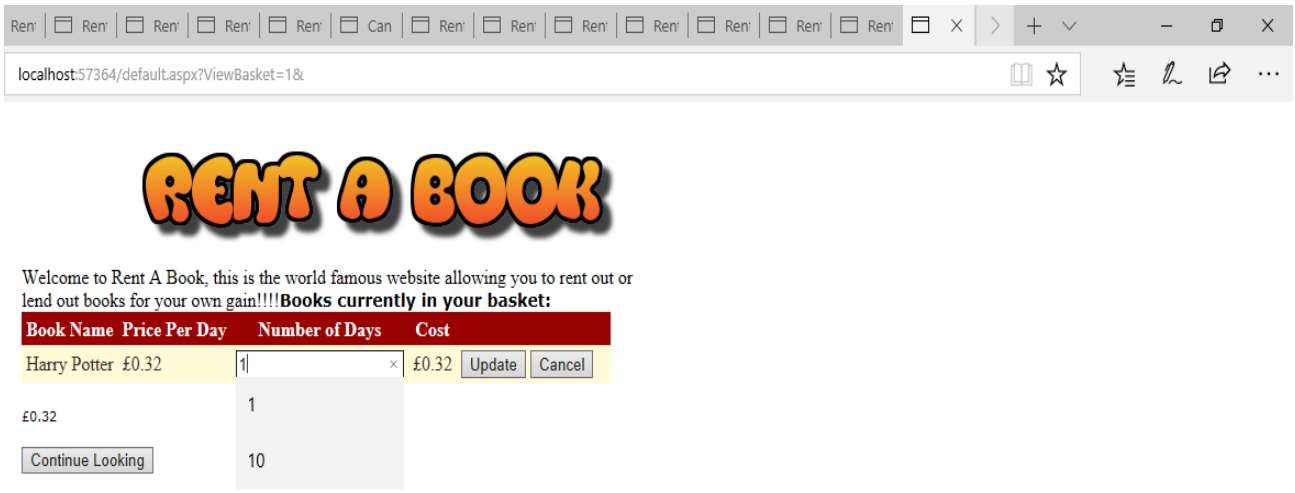


Image 7:

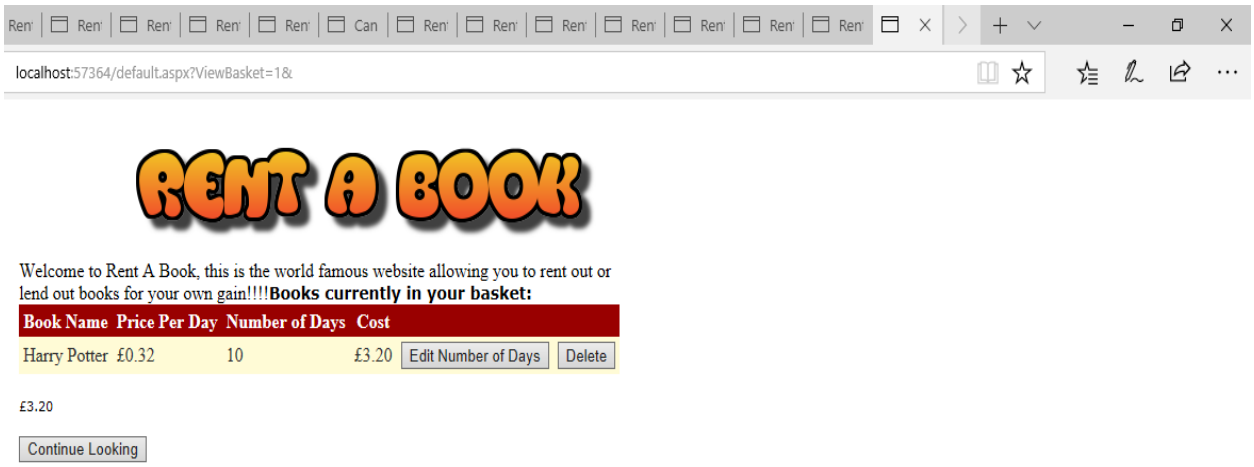
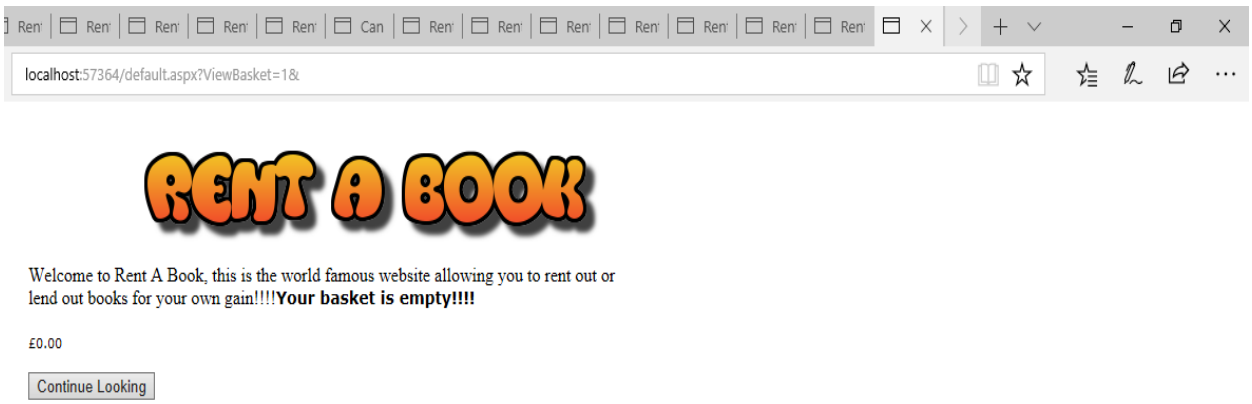


Image 8:



Review

In this Iteration I have been able to create a working search bar that is able to search for a book by querying through the names and description. It then gives points for if it is found in the name or description, resulting in the books closest to what the user wants to be shown first. I have also been able to successfully make a basket so the user is now able to add books to their own personal basket, remove books and update the number of days they want the book for.

Stakeholder review

User Interface

The site is definitely taking shape, the new text search is great and works well. The site is uncluttered and it's obvious to a new user what the purpose of the site is. The pages are bright and colourful which I believe will give a better user experience.

Functionality

The new basket functionality is fantastic and is very familiar to any user who has bought an item on the web. I've checked adding and removing books, both of which worked well.

Having tested the search facility, I'm very impressed with the results I'm seeing. I've done several text searches and not only does it search for book titles, but also authors names and even descriptions containing the text. I can see this being very useful for people.

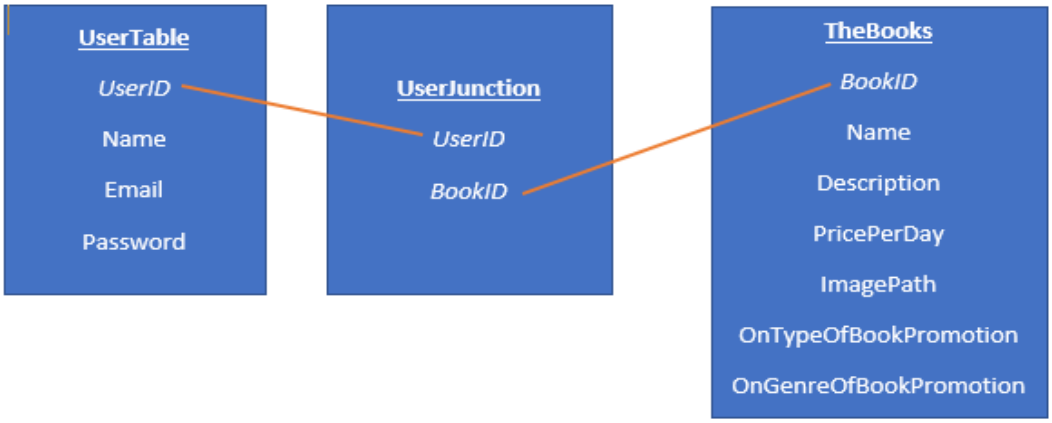
Iteration 3 - *Date 2/12/17*

Aims for this iteration

The aims for this prototype is to have the website to allow the users to create an account with it. The account must contain the name, email and password of the user. The website must also have a button that allows the user to log in and add a book onto the website. When adding the book, the user must fill in the book's name, description, price per day, genre and upload an image of it.

Functionality that the prototype will have

For this prototype to work I will need to add two more new tables into the database. One is to hold the user's information, while the other is a userJunction. The userJunction is needed because I need to create a relationship between the books and the users, plus one user may own multiple books on the website so it also allows for a one to many relationship.



Annotated code screenshots with description

Making the user tables and sign up pages

To make a signup area on the website I first needed to make a table to hold all the users' information. To do this I right clicked on tables in the RentABook database and selected Add New

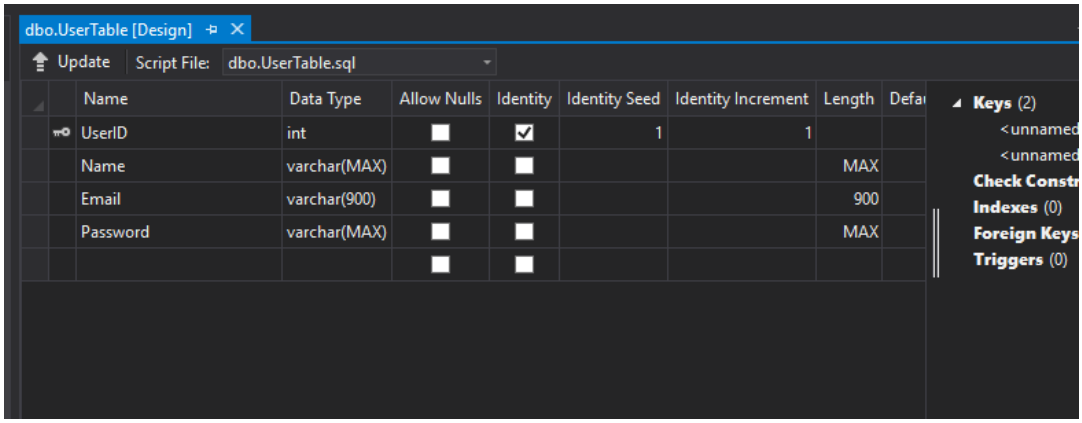
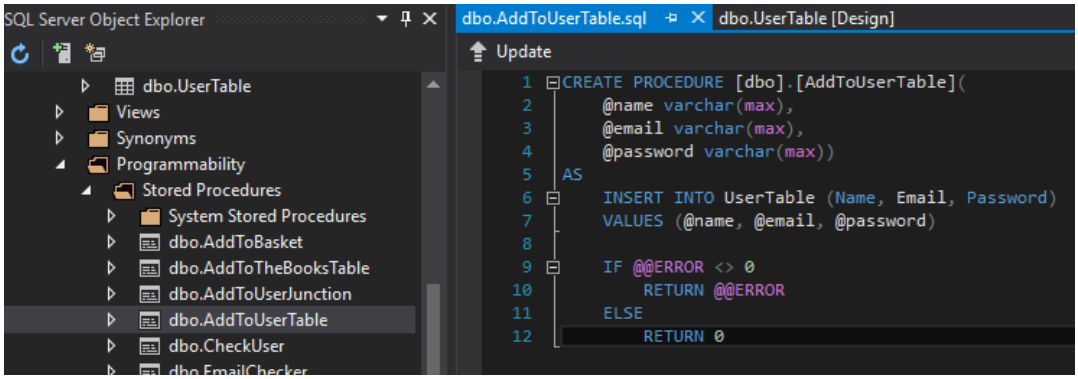
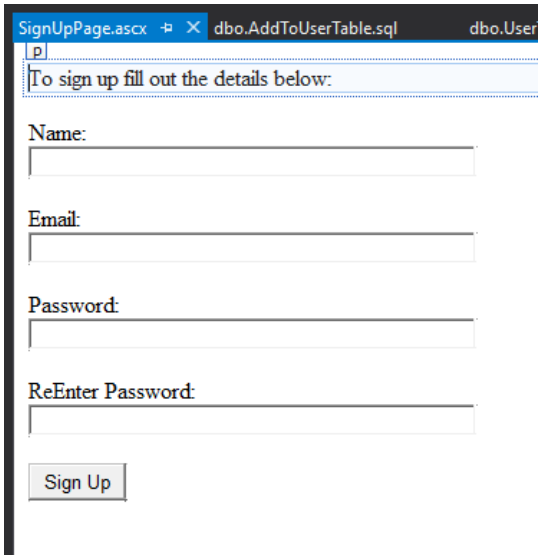


Table. The table will contain four columns, UserID, Name, Email and Password. I have set UserID to be the primary and also the identity key. This means that every single user in the table will have a unique identifier even if tyey share the same name. After that I then made the SQL stored procedures relating to it. The first stored procedure I made was the AddToUserTable procedure. The procedure takes in three



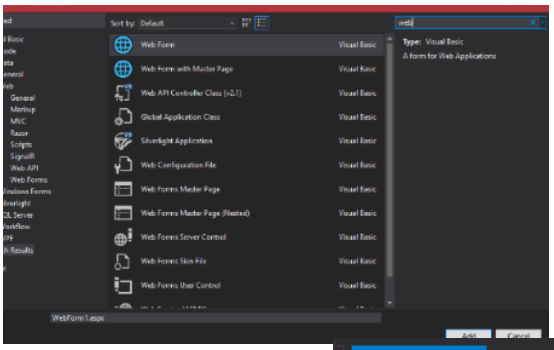
parameters, the name of the person, their email and their password.the procedure inserts this information about the user into a UserTable, if there is an error, the procedure will return out the error.



After that I right clicked on solution explorer and clicked on Add where I selected web forms user control. I then named the user control to SignUpPage, and went onto design view and edited it to this. I have added five lables each stateing what to type in the box below it, four textboxes for the user to input their answers and a button called Sign Up. Afterwords I went onto the code behind file, SignUp.ascx.vb, and set it so that when the user clicks on the sign up button

```vb
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

End Sub

Protected Sub SignUpButton_Click(sender As Object, e As EventArgs) Handles SignUpButton.Click
    Dim name As String = NameTextBox.Text
    Dim email As String = EmailTextBox.Text
    Dim password As String = PasswordTextBox.Text
    Dim rePassword As String = ReEnterPasswordTextBox.Text

    If password <> rePassword Then
        SignUpLabel.Text = "Passwords do not match, please re-enter"
    Else
        Try
            Account.AddUser(name, email, password)
            SignUpLabel.Text = "You have signed up!"
            hide()
        Catch ex As Exception
            SignUpLabel.Text = "This email address is already in use!"
        End Try
    End If
End Sub

Private Sub hide()
    NameLabel.Visible = False
    NameTextBox.Visible = False
    EmailLabel.Visible = False
    EmailTextBox.Visible = False
    PasswordLabel.Visible = False
    PasswordTextBox.Visible = False
    ReEnterPasswordLabel.Visible = False
    ReEnterPasswordTextBox.Visible = False
    SignUpButton.Visible = False
End Sub
End Class
```

it will first set the variables name, email, password and rePassword to hold the corrisponding data from the textboxs. It will then compare password to rePassword and if they do not equal each other then it informs the user that their passwords do not match. However if the passwords do match then it will call the function AddUser from the Account class, it will then inform the user that they have signed up and it will hide all of the textboxes, lables and buttons on the page. If the code catches an exception thrown at it then it will tell the user that the email address they used is already in use.



Next, I needed to make another Web Form by right clicking solution explorer, then clicking on Add and selecting Web Form. I then named the file Accounts.vb.

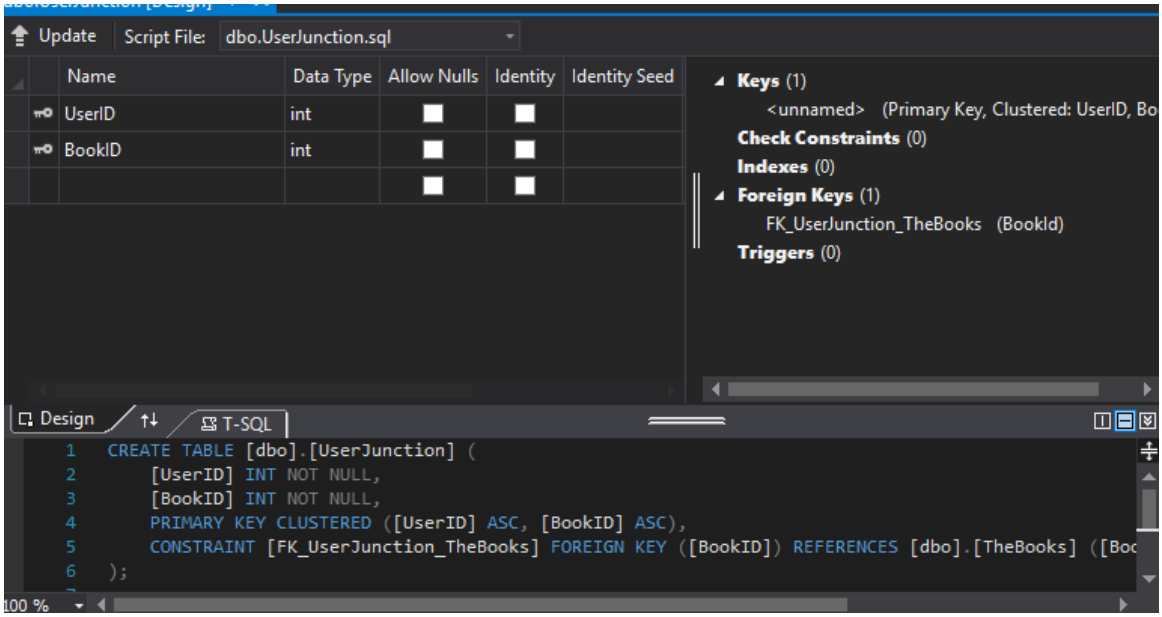
In this file, I made the Adduser function which can take in the parameter Name, Email and Password. The function will then execute the SQL stored procedure

```vb
Imports System.Data.SqlClient

Public Class Account

    Private Shared ReadOnly Property connectionString() As String
        Get
            ' Return ConfigurationSettings.AppSettings("ConnectionString")
            Return System.Configuration.ConfigurationManager.AppSettings("ConnectionString")

        End Get
    End Property

    Public Shared Function AddUser(ByVal Name As String, ByVal Email As String, ByVal Password As String)
        Dim connection As New SqlConnection(connectionString)
        Dim command As New SqlCommand("AddToUserTable", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@name", SqlDbType.Char, 36)
        command.Parameters("@name").Value = Name
        command.Parameters.Add("@email", SqlDbType.Char, 36)
        command.Parameters("@email").Value = Email
        command.Parameters.Add("@password", SqlDbType.Char, 36)
        command.Parameters("@password").Value = Password
        Try
            connection.Open()
            command.ExecuteNonQuery()
        Finally
            connection.Close()
        End Try
        Return 0
    End Function
```

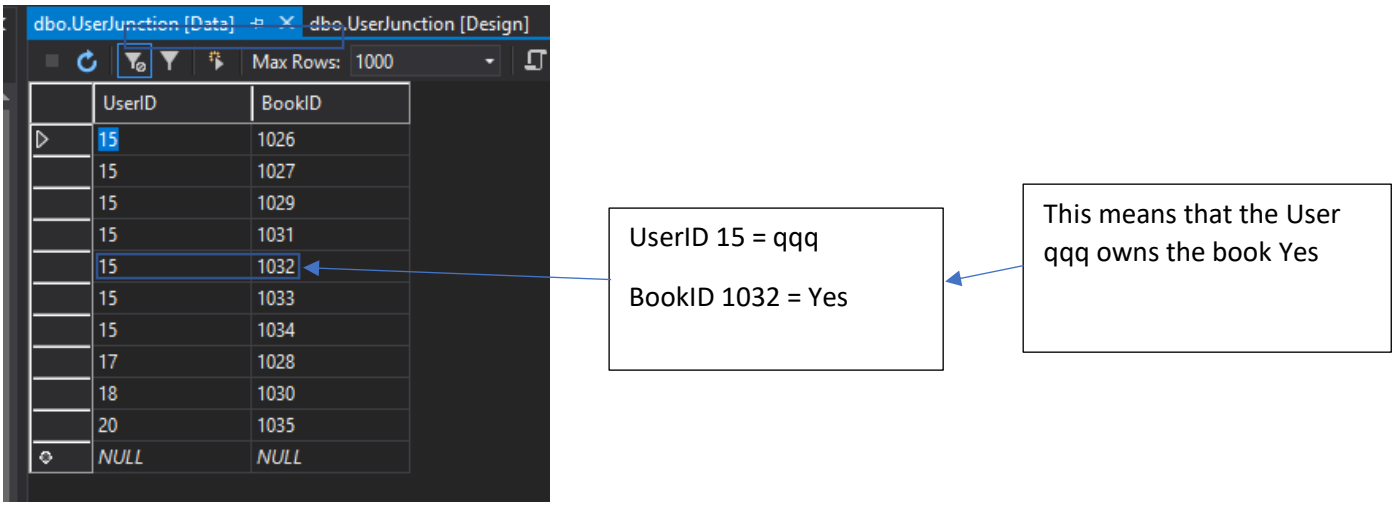
called AddToUserTable. This procedure takes in the Name, Email and password as its parameters.

Allowing the user to add the books

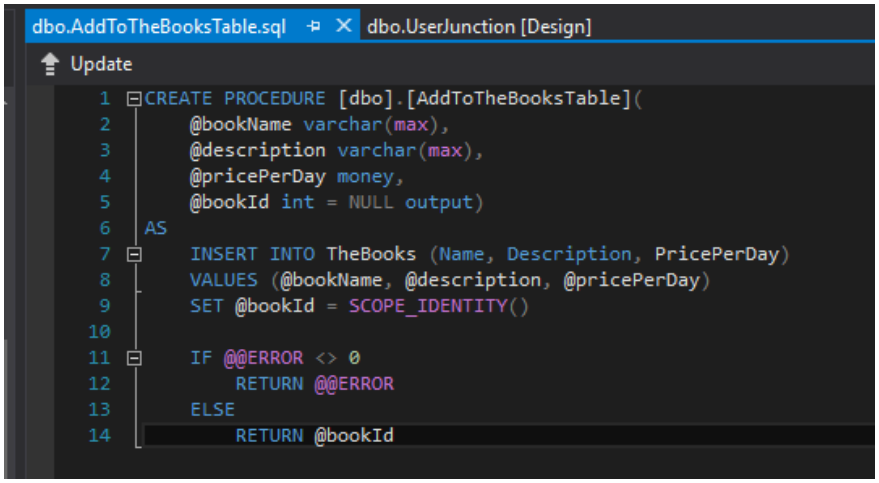
To allow the user to add the books, I needed a UserJunction table. This UserJunction table will do the same thing as the BookJunction table as the UserJunction table creates a many to many relationship



between the tables, TheBooks and the UserTable. This is needed because a single user can own multiple books and it is the only way of connecting the books to the users. The UserJunction table contains two columns, the first is the UserID while the other is the BookID, both being set to be foreign keys, creating the connection.

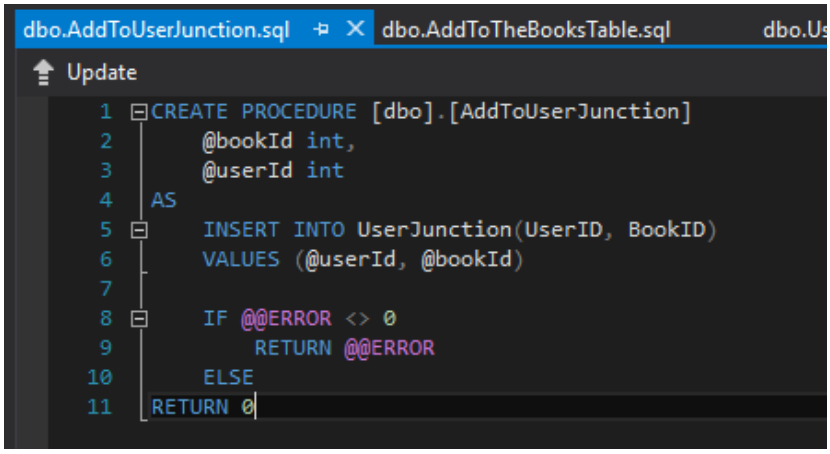


After creating the table I then needed to create the SQL stored procedure to add the books to the books table. To do this I right clicked on stored procedures and added a new procedure, naming it



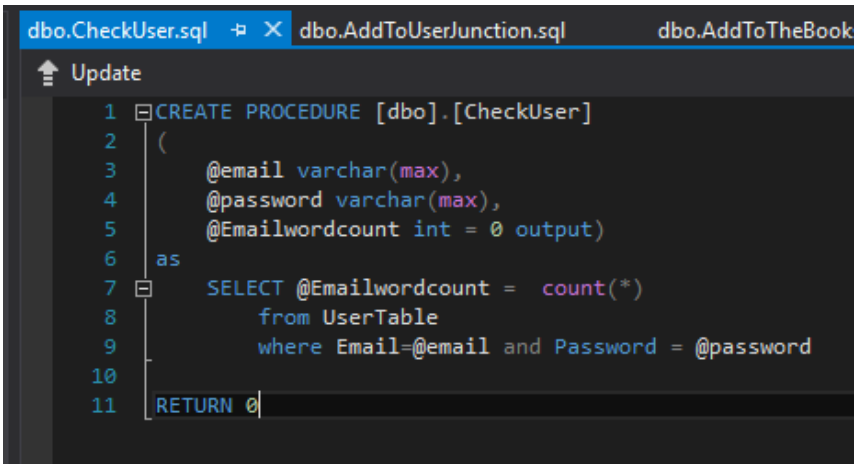
AddToTheBooksTable. The procedure takes in three parameters, bookName, description, PricePerDay. The procedure will then insert the bookName, description and PricePerDay into the TheBooks table. If there is an error with adding it will return error, otherwise it will return the new bookId.

The next stored SQL procedure was the AddToUserJunction procedure. This procedure adds the



parameters bookId and userId to the UserJunction table using the insert into command which adds it to the back of the table. If there is an error with adding it will return error instead of 0.

Another procedure I needed to make was the Check User procedure. This procedure will search through the UserTable to check is it exists in there or not. It does this by taking in the two parameters, email and password. It will then count how many times the email the user used exist in the table, in the

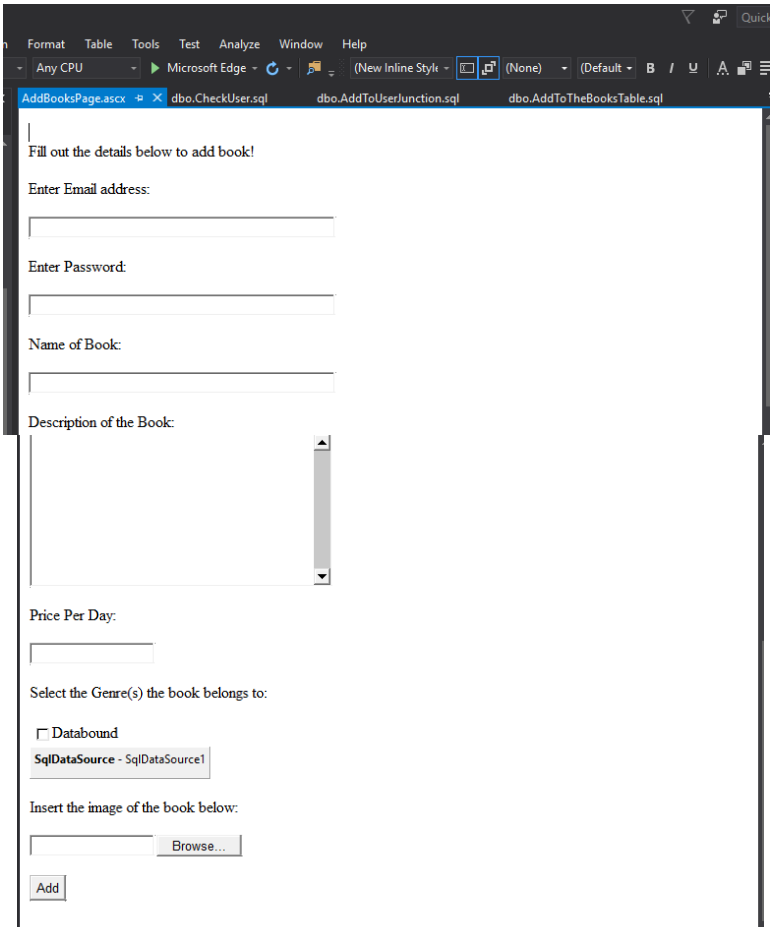


column email. The procedure will then return the number of results found.

After making the SQL stored procedures, I then moved on to making a web forms user control called

AddBooksPage.ascx. next I went onto design view and dragged and dropped eight labels, to explain to the user what to put in the textboxes or the check box list. I then connected the checkbox list to the GenreOfBookTable by right clicking it setting its source. After that I added the Add button at the end of the screen for the user to click once finished. I then accessed

```
AddBooksPage.ascx.vb | AddBooksPage.ascx | dbo.CheckUser.sql | dbo.AddToUserJunction.sql
Rent A Book                                  AddBooksPage
1   Public Class AddBooksPage
2       Inherits System.Web.UI.UserControl
3
4       Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
5
6       End Sub
7
8       Protected Sub AddButton_Click(sender As Object, e As EventArgs) Handles AddButton.Click
9           Dim Email As String = EmailTextBox.Text
10          Dim Password As String = PasswordTextBox.Text
11          Dim BookName As String = NameOfBookTextBox.Text
12          Dim Description As String = DescriptionOfTheBookTextBox.Text
13          Dim PricePerDay As String = PriceTextBox.Text
14          Dim SelectedGenres As ListItem = GenreCheckBoxList.SelectedItem
15          'Dim Image As Image = ImageUpload.
16          Dim authentic As Integer = Account.CheckUser(Email, Password)
17
18          If authentic <> 1 Then
19              AddBookLabel.Text = "Your Email or Password is incorrect!"
20          Else
21              Dim BookId As Integer = Account.AddBooktoTheBooksTable(BookName, Description, PricePerDay)
22              Account.LinkUserAndBook(BookId, Email)
23          End If
24
25
26      End Sub
27  End Class
```



the code behind file of the AddBooksPage.ascx and added a new piece of code which is ran if the Add button is clicked. It will first check to see if your email and password matches with one of the already signed up accounts. It does this by calling the CheckUser function from the class Accounts and saving the results to authentic. If authentic does not equal 1 then it will inform the user that they have entered the wrong password or email. Otherwise it will call the functions AddBookToTheBooksTable and LinkUserAndBooks from the Accounts class.

The next thing I needed to do was make the three functions, the CheckUser function, AddBookToTheBooksTable function and the LinkUserAndBooks function, in the account class. The function takes in two parameters, email and password, it will then create a connection to the SQL stored procedure called CheckUser. After establishing a connection, it will execute it and pass the parameters, email and password, into it. The

```
    Public Shared Function CheckUser(ByVal Email As String, ByVal Password As String)
        Dim connection As New SqlConnection(connectionString)
        Dim command As New SqlCommand("CheckUser", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@email", SqlDbType.Char, 36)
        command.Parameters("@email").Value = Email
        command.Parameters.Add("@password", SqlDbType.Char, 36)
        command.Parameters("@password").Value = Password
        command.Parameters.Add("@Emailwordcount", SqlDbType.Int)
        command.Parameters("@Emailwordcount").Direction = ParameterDirection.Output

        Dim authentic As Integer = 0
        Try
            connection.Open()
            command.ExecuteNonQuery()
            authentic = command.Parameters("@Emailwordcount").Value
        Finally
            connection.Close()
        End Try
        Return authentic
    End Function
```

procedure will then return out @emailWordCount which will contain either 1 or 0. The function will save this to the variable authentic which is set to be an integer, the function also returns authentic out, to be used in the code behind file of AddBooksPage.ascx.

The function, AddBookToTheBooksTable function. This function will be used to execute the AddToTheBooksTable stored SQL procedure and passes in the parameters Name, Description and PricePerDay. Before the function closes the connection the stored procedure

```
    Public Shared Function AddBooktoTheBooksTable(ByVal Name As String, ByVal Description As String, ByVal PricePerDay As String)
        Dim connection As New SqlConnection(connectionString)
        Dim command As New SqlCommand("AddToTheBooksTable", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@bookName", SqlDbType.Char, 36)
        command.Parameters("@bookName").Value = Name
        command.Parameters.Add("@description", SqlDbType.Char, 36)
        command.Parameters("@description").Value = Description
        command.Parameters.Add("@pricePerDay", SqlDbType.Char, 36)
        command.Parameters("@pricePerDay").Value = Convert.ToDouble(PricePerDay)
        command.Parameters.Add("@bookId", SqlDbType.Int, 36)
        command.Parameters("@bookId").Direction = ParameterDirection.Output
        Dim bookId As Integer = 0
        Try
            connection.Open()
            command.ExecuteNonQuery()
            bookId = command.Parameters("@bookId").Value
        Finally
            connection.Close()
        End Try
        Return bookId
    End Function
```

returns out the new bookId for the added book, the bookId is then returned outfrom the function.

The final function I needed to make was the LinkUserAndBook function. When this function is called two parameters are passed into it. The first one is the new book's ID, while the other is the user's ID.

```
    Public Shared Function LinkUserAndBook(ByVal BookId As Integer, ByVal Email As String)
        Dim userID As Integer = getUserId(Email)
        Dim connection As New SqlConnection(connectionString)
        Dim command As New SqlCommand("AddToUserJunction", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@bookId", SqlDbType.Int, 36)
        command.Parameters("@bookId").Value = BookId
        command.Parameters.Add("@userId", SqlDbType.Char, 36)
        command.Parameters("@UserId").Value = userID
        Try
            connection.Open()
            command.ExecuteNonQuery()
        Finally
            connection.Close()
        End Try
        Return 0
    End Function
```

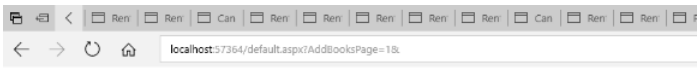
The function then creates a connection to the AddToUserJunction stored SQL procedure. Once a

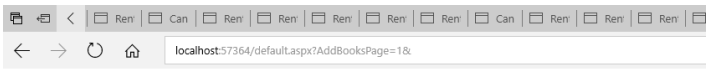
connection has been secured it executes the procedure and gives it the parameters needed, the user's ID and the book's ID. After the procedure is finnished the function closes down the connection between them.

The final thing I needed to do now was to update the code behind file of default.ascx. I first added a new if statement which will check to see if the user has clicked the add book button. If they havent it

```
default.aspx.vb | default.aspx | Account.vb | AddBooksPage.ascx.vb | AddBooksPage.ascx | dbo.CheckUser.sql
Rent A Book                                  _default

3
4       Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
5           Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
6           Dim searchString As String = Request.QueryString("Search")
7           Dim ViewBasket As String = Request.QueryString("ViewBasket")
8           Dim SignUp As String = Request.QueryString("SignUp")
9           Dim AddBooksPage As String = Request.QueryString("AddBooksPage")
10          If Not AddBooksPage Is Nothing Then
11              Dim control As Control
12              control = Page.LoadControl("UserControls/AddBooksPage.ascx")
13              pageContentsCell.Controls.Add(control)
14          ElseIf Not SignUp Is Nothing Then
15              Dim control As Control
16              control = Page.LoadControl("UserControls/SignUpPage.ascx")
17              pageContentsCell.Controls.Add(control)
18          ElseIf Not ViewBasket Is Nothing Then
19              Dim control As Control
20              control = Page.LoadControl("UserControls/Basket.ascx")
21              pageContentsCell.Controls.Add(control)
22          ElseIf Not searchString Is Nothing Then
23              Dim control As Control
24              control = Page.LoadControl("UserControls/SearchResults.ascx")
25              pageContentsCell.Controls.Add(control)
26          ElseIf Not TypeOfBooksID Is Nothing Then
27              Dim control As Control
28              control = Page.LoadControl("UserControls/Catalog.ascx")
29              pageContentsCell.Controls.Add(control)
30          Else
31              Dim control As Control
32              control = Page.LoadControl("UserControls/FirstPage.ascx")
33              pageContentsCell.Controls.Add(control)
34          End If
100 %
```

will move onto the next elseif statement, but if they have the program will then add AddBooksPage.ascx to the pageContentsCell.





RENT A BOOK

Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Fill out the details below to add book!

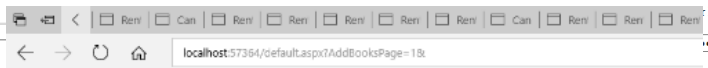
Enter Email address:

Enter Password:

Name of Book:

Description of the Book:

Price Per Day:



RENT A BOOK

Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Your Email or Password is incorrect!

Enter Email address:

Braniff007@gmail.com

Enter Password:

Name of Book:

James Bond

Description of the Book:

The name's Bond, James Bond

Price Per Day:

RENT A BOOK

Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Fill out the details below to add book!

Enter Email address:

Braniff007@gmail.com

Password:

••••••••

of Book:

s Bond

ription of the Book:

name's Bond, James Bond

Per Day:

Test Results / Evidence

What is being tested	***Input***	***Justification of input***	***Expected outcome***	***Actual outcome***	***How to solve***	***Proof***
Sign up page is displayed	Sign up button is clicked	Clicking the signup button is the only way for the user to enter the page	Signup page to be displayed	Sign up page was displayed	N/A	Image 1
Sign up successful	Fill out the details on the page	This will check to see if the user's info is saved to the user table	The website informs the user that it has been saved	The webpage informed that the info was saved	N/A	Image 2 to 3
Add book page displayed	The add book button is clicked	This will check if the page is able to load after the designated button is clicked	The add book page should be displayed	The add book page was displayed	N/A	Image 4
Book added	Fill out the details on the page	This will check to see if the book is saved to the database	Should be able to search for it after added	The newly added book was found when searched for	N/A	Image 5 to 6

Image 1:

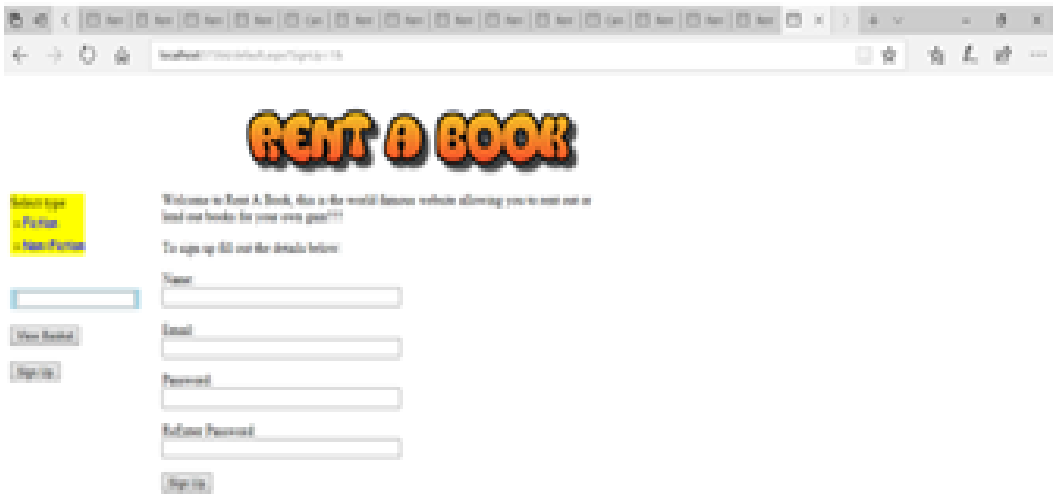


Image 2:

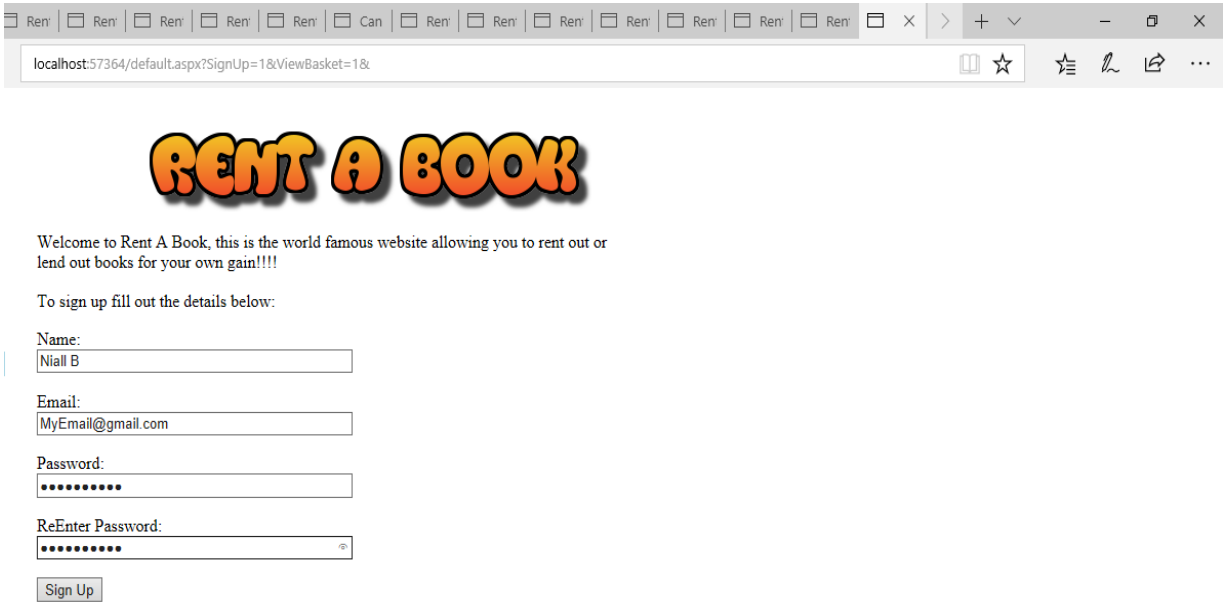


Image 3:

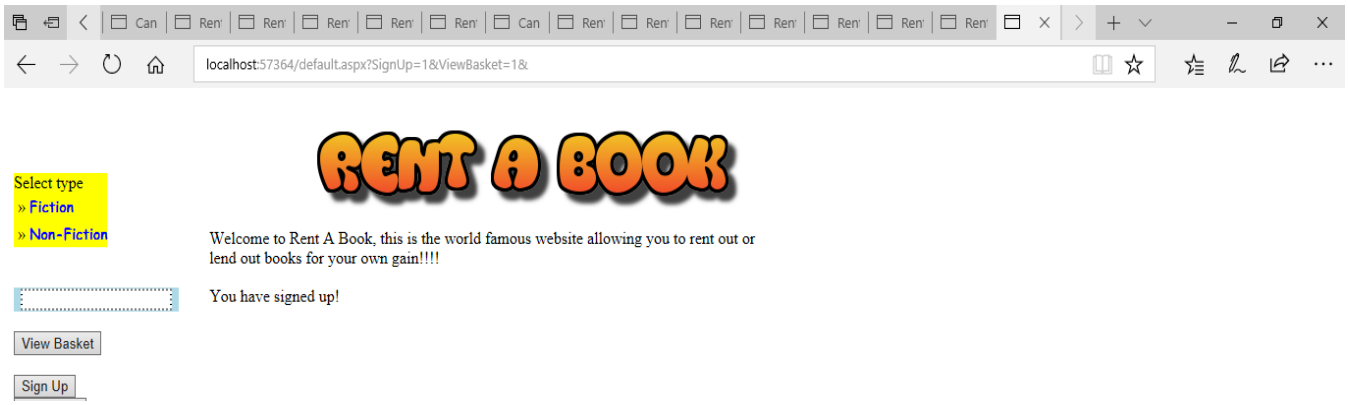


Image 4:

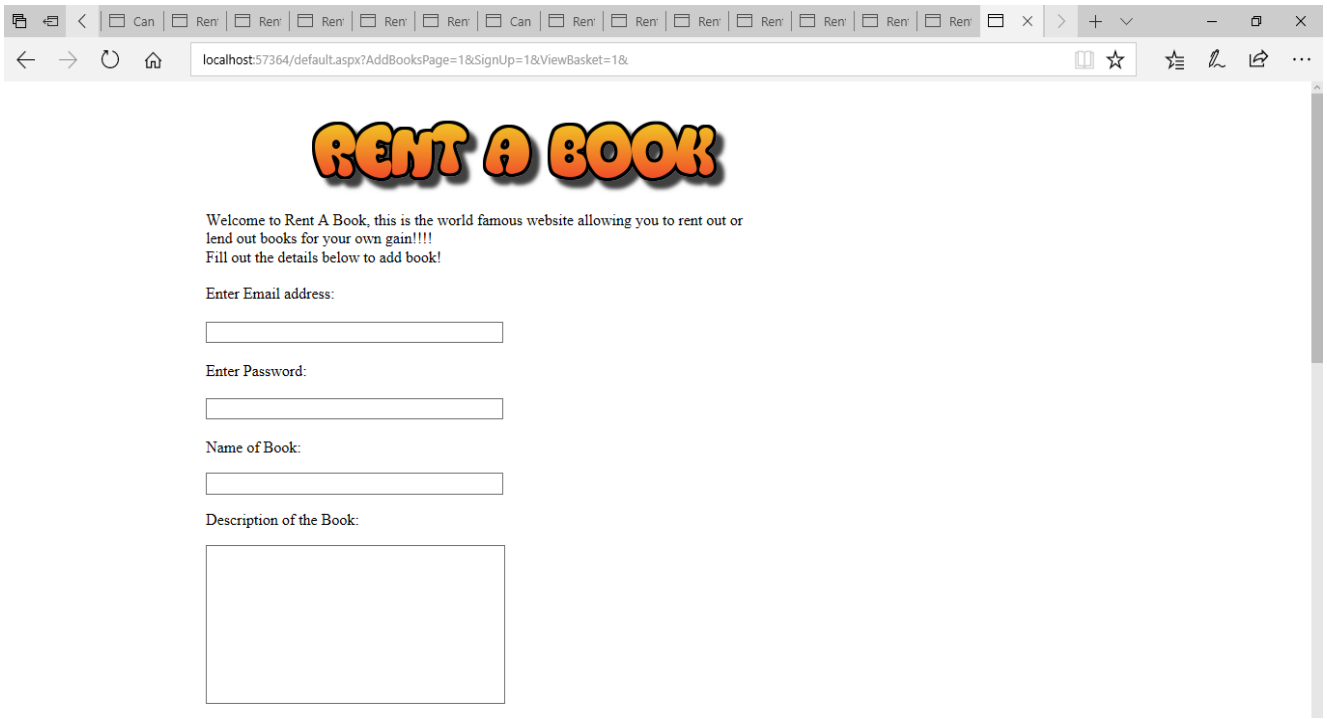


Image 5:

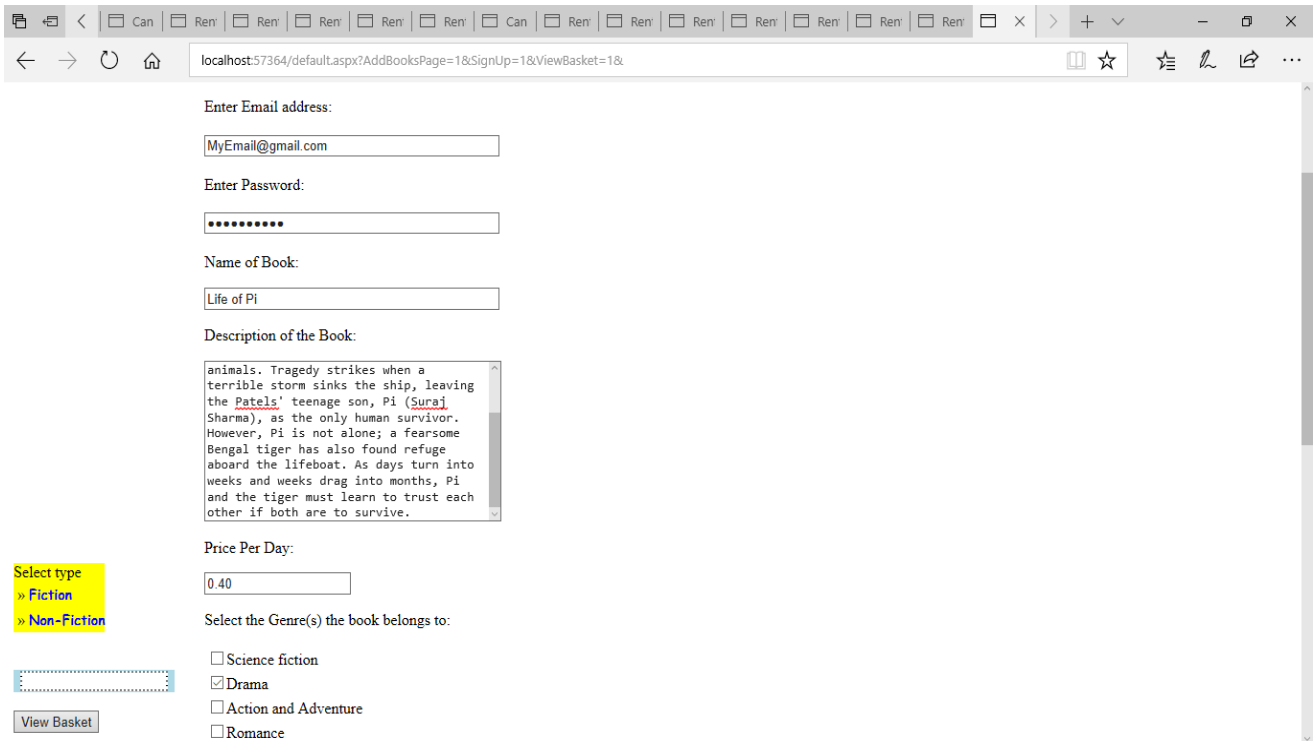
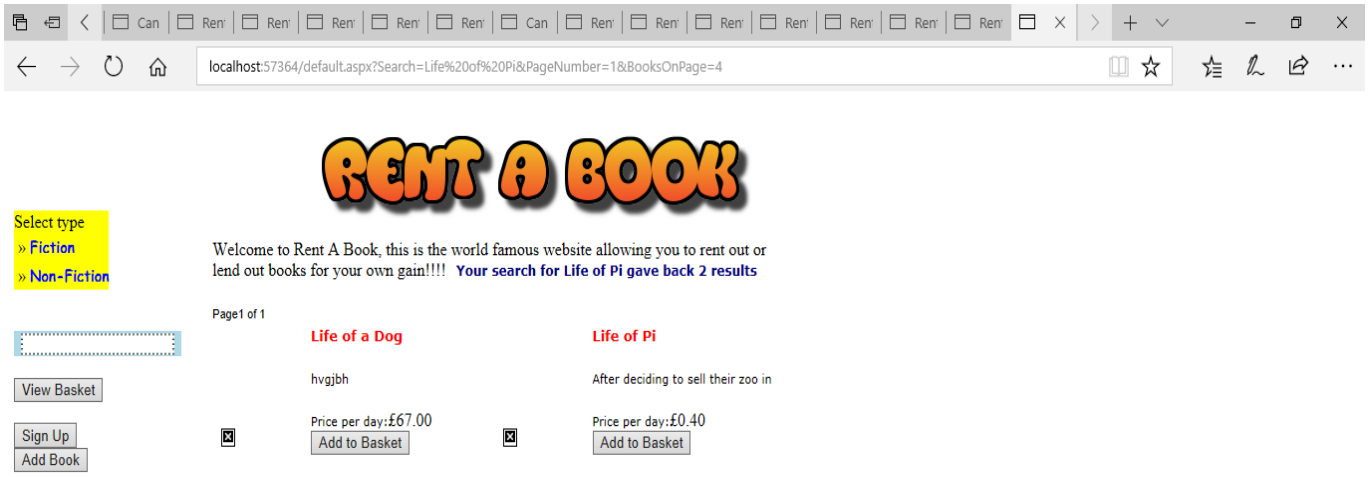


Image 6:



Review

This iteration has been successful as I was able to create a signup form that links to the user table and allows the user to save their details onto it, therefore creating an account with the website. I was then able to create another form that allows them to type up the necessary details for adding the book, this form also asked for their username and password, so I did not need to create a separate form for this. The book is then successfully added and is linked up with the owner as well. The book can also be found when searched for in the search bar.

Stakeholder review

User Interface

All the pages look good now. They are clear and easily navigated which gives a great user experience. I like that they all have a similar look and feel, with the company logo on the top of each page. The sign-up page has now been added and the ability for users to add new books which they want to lend.

Functionality

The sign-up page is exactly what was asked for and is not overly onerous for users to sign-up. Importantly, it records peoples email addresses and requires a password to ensure that only authorised users can make changes.

The new functionality to add new books to the system works well and is very easy to use. This should encourage users to add more books and therefore help grow the popularity of the site.

Testing: Evaluative

Final Testing Evidence: Functionality and Robustness

For robustness testing I entered in multiple boundary answers and invalid answers. This was to test to see if the website would accept the invalid data responses and crash, or if it popups with an alert telling the user that they have entered in an invalid response and they need to try again. These tests were performed during the black box testing and are highlighted with a  The boundary testing that was done was only done for when the user entered in the number of days they wish to rent the book out for. This is because the number of days is the only input field that has a minimum number. I tried crashing the program by entering in incorrect email addresses and passwords. This was to make sure if the users accidently did this the program will not crash, but instead alert them of this error.

Black box testing

Test No.	Aspect being tested	Justification	Test Data	Type of test (valid, invalid, boundary)	Expected result	Actual result	Evidence (screenshot, screencast etc)	Successfull met
1	Does home page load	This is the main page and is the first page the customer will see when they go onto the website. This page will also allow the user to find and rent the books.	Open the website	Valid	Homepage is to load	Home page has loaded when opened	Image 1	Fully successful
2	Is the customer able to search for	Fiction and non-fiction are the two types of books, the list of	Fiction is selected	Valid	The corresponding list of genres to fiction appear	The correct list of genres and books appeared when the user selected fiction	Image 2 Also shown in screencast	Fully successful

	a book by genres	genres will then appear underneath the type selected for the user to choose			along with fictional books		video at 00:23	
3			Action and adventure is selected	Valid	Action and adventure books are only shown	The correct genre books appeared	Image 3 Also shown in screencast video at 00:26	Fully successful
4			Non-fiction is selected	Valid	The corresponding list of genres to non-fiction is to appear along with non-fictional books	The correct list of genres and books appeared when the user selected non-fiction	Image 4 Also shown in screencast video at 00:08	Fully successful
5			Maths is selected	Valid	Maths books are only shown	The correct genre books appeared	Image 5 Also shown in screencast video at 00:14	Fully successful
6	Is the customer able to view their own basket	This is an important feature as it must work correctly for the customer to see what they are about to rent.	View basket button is clicked	Valid	The user's personal basket to appear	The user's basket has appeared along with one of their books, Harry Potter	Image 6 Also shown in screencast video at 00:33	Fully successful

7	Is the customer able to add a book to their own basket	This is another important feature as the website must enable the customer to choose what they want to rent by adding it to their basket.	Add to basket button clicked for Maths book	Valid	Maths book should appear in the basket	The maths book has appeared in the user's basket when add to basket has been clicked underneath the book.	Image 7 Image 8 Also shown in screencast video at 00:19	Fully successful
8			Add to basket button clicked for Harry Potter book	Valid	Harry Potter book should appear in the basket	The Harry Potter book has appeared in the user's basket when add to basket has been clicked underneath the book.	Image 9 Image 10 Also shown in screencast video at 00:30	Fully successful
9	Is the customer able to update the number of days they wish to rent for	This must work to allow the customer to choose how long they want to rent out a book for	Change number of days to 10 for Maths book	Valid	Number of days to change from 1 to 10 for the maths book in the basket	When the user clicked on the Edit number of days button, they could change the number of days to 10 and update the basket with it.	Image 11 Image 12 Image 13 Also shown in screencast video at 00:38	Fully successful
10			Change number of days to 1 for maths book	Boundary	Number of days to change from 10 to 1 for the maths book in the basket	When the user clicked on the Edit number of days button, they could change the number of days	Image 13 Image 14 Image 15	Fully successful

						to 1 and update the basket with it.		
11			Change number of days to -10 for Maths book	Invalid	An error report should appear telling the customer that the number of days entered is invalid	An error appeared saying invalid number of days	Image 15 Image 16	Fully successful
12	Is the customer able to delete a book from their own basket	The customer must be able to remove books they no longer want to rent out	Clicking the delete button for maths book	Valid	The maths book is removed from the basket	The maths book is removed the basket after the delete button is clicked	Image 17 Image 18 Also shown in screencast video at 00:48	Fully successful
13	Is the customer able to rent a book from the website?	This is the main function of my solution and must enable the user to rent out a book	Rent Harry Potter by clicking on Rent button	Valid	Email of confirmation is sent to customer with transfer of money and book	Confirmation email was sent to the customer, but the transfer of money and book need to be done between the owner and customer	Image 29 Image 30 Image 31 Also shown in screencast video at 03:47	Partially successful
14	Is the customer able to use their	This is a common method for users to pay	Pay for Harry Potter book	Valid	Payment successful	Payment unsuccessful. This was because I did	N/A	Not successful

	PayPal account to pay for the rent of a book	for things online, which is why it is a handy feature for the customer to have.	using PayPal			not have enough time to implement PayPal transactions into my program		
15	Is the customer able to pay for the rent of the book	Another payment method would need to work in case the use does not have PayPal	Pay for Harry Potter book using secondary method	Valid	Email with contact details of customer is sent to the owner of the book	Contact details of the customer was sent to the owner of the book successfully for them to arrange payment	Image 31	Fully successful
16	Is the customer able to use predictive search to help them find a book	Predictive search will help the customer find books by showing them titles of books like what being typed in	Start typing in search box until books are suggested underneath it	Valid	Book titles to appear under search bar when typing	I partially complete this feature as it can show previous searches when typing	Image 19 Also shown in screencast video at 03:26	Partially successful
17	Is the customer able to use suggestive search to find books they may like	This feature is handy as it enables the user to view books they may like but never thought of before.	When having searched for something, the website will then suggest some other books based on what you	Valid	A list of suggested books to be shown after a search	I was unable to meet this as I ran out of time	N/A	Not successful

			have searched					
18	Is the customer able to sign up to the website	This is another important feature the website must be able to do as it enables the customer to sign up and rent out books	Type in an invalid email when signing up	Invalid	Error should appear to tell user that sign up has failed	Error appeared telling the user that sign up has failed	Image 26 Also shown in screencast video at 00:56	Fully successful
19			Type in two different passwords	Invalid	Error should appear to tell user that sign up has failed	Error appeared telling the user that sign up has failed	Image 20	Fully successful
20			Type in an email address that already exists in the database	Invalid	Error should appear to tell user that sign up has failed	Error appeared telling the user that sign up has failed	Image 21 Also shown in screencast video at 01:25	Fully successful
21			Fill out signup form correctly	Valid	Sign up completed	Successfully signed up	Image 22 Also shown in screencast video at 01:57	Fully successful
22	Is the customer able to log into	The is also an important feature to my solution as the	Type in the wrong password	Invalid	Error should appear to tell user that they have either entered the email or	Error appeared saying that email or password was incorrect	Image 23	Fully successful

	their account	customer must be able to log into their account to add books to the website and to rent books out.	when login in		password in wrong and didn't log in			
23			Enter in the wrong email address when login in	Invalid	Error should appear to tell user that they have either entered the email or password in wrong and didn't log in	Error appeared saying that email or password was incorrect	Image 24	Fully successful
24			Enter in correct email and password	Valid	Login successful	Successfully logged in	Image 27	Fully successful
25	The customer should be able to use the read list to keep track of the books they wish to rent	This is not an essential feature to my solution but can make it easier for the user to keep track of the books they want to read.	Add Harry Potter book to read list	Valid	Successfully add the Harry Potter book to read list	Harry Potter was added to the read list	Image 32 Image 33	Fully successful
26			View read list	Valid	Able to see read list with Harry Potter book inside it	When read list was opened, The Harry Potter book was	Image	Fully successful

						contained inside.		
27			Remove Harry potter book from read list	Valid	Successfully remove Harry Potter from the list	Harry potter was successfully removed from the read list	Image	Fully successful
28	The customer must be able to add their own books to the website, for other customers to rent out	This is an essential feature for my solution as the website must allow the customers to rent books from one another, meaning they must be able to put their book up onto the website.	Fill out Add book form	Valid	Their Book which they added should be found when searched for	The book was found when searched for after it was added.	Image 25 Also shown in screencast video at 02:01	Fully successful

Image 1

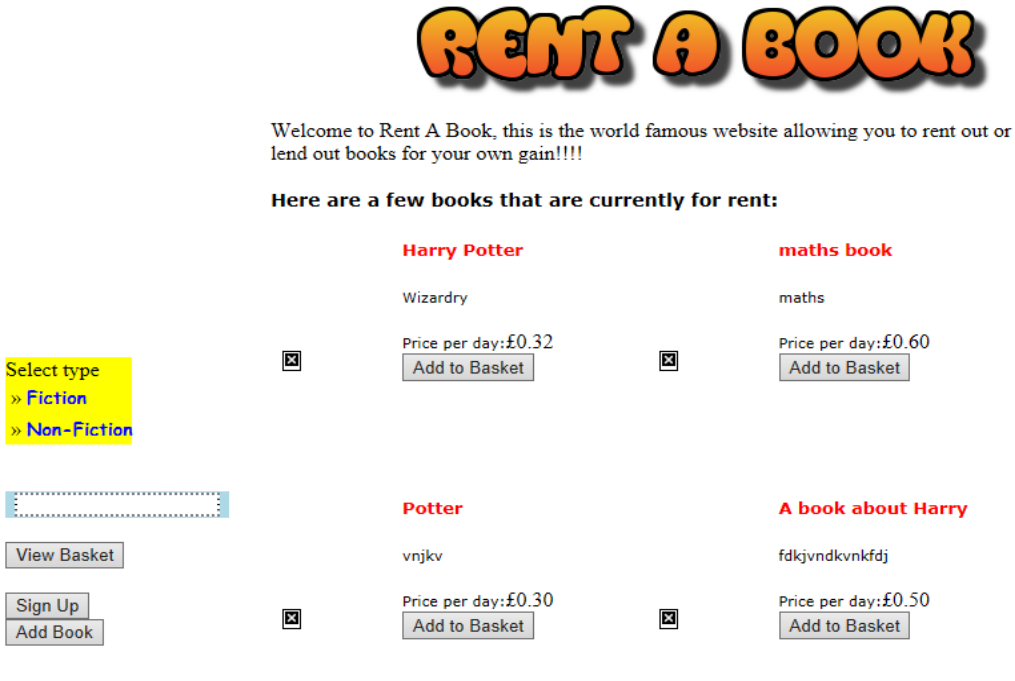


Image 2

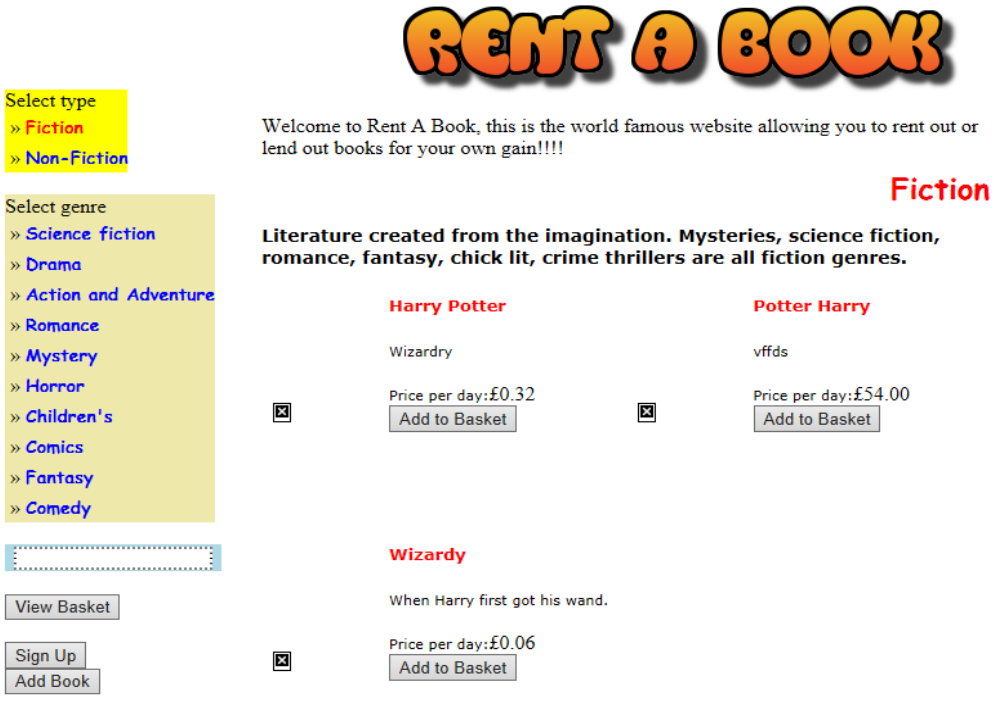


Image 3

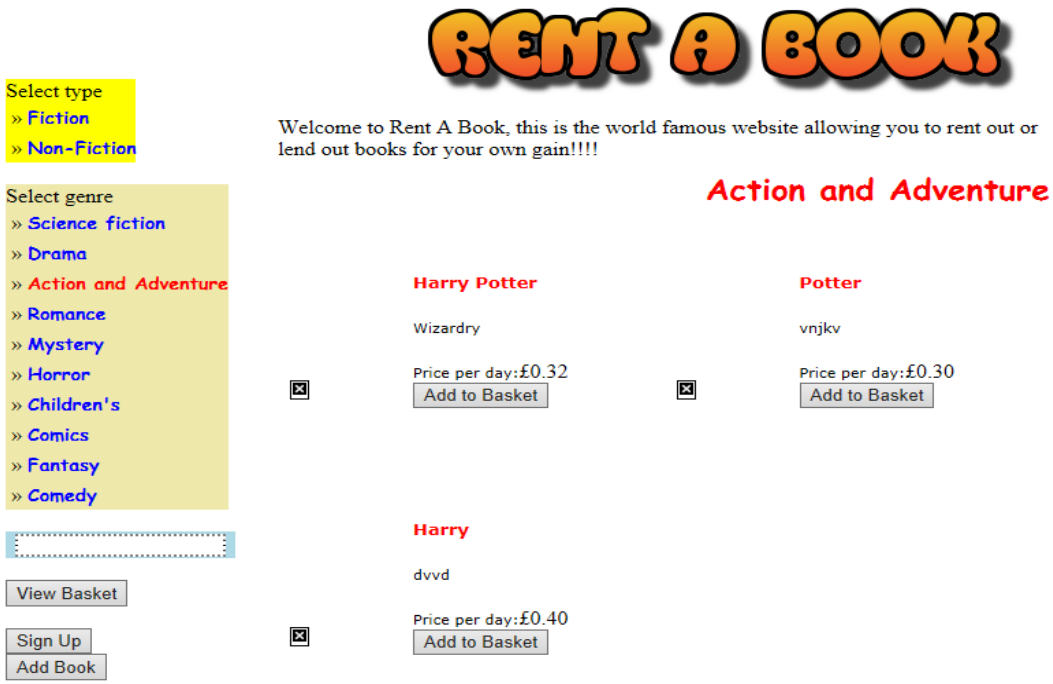


Image 4

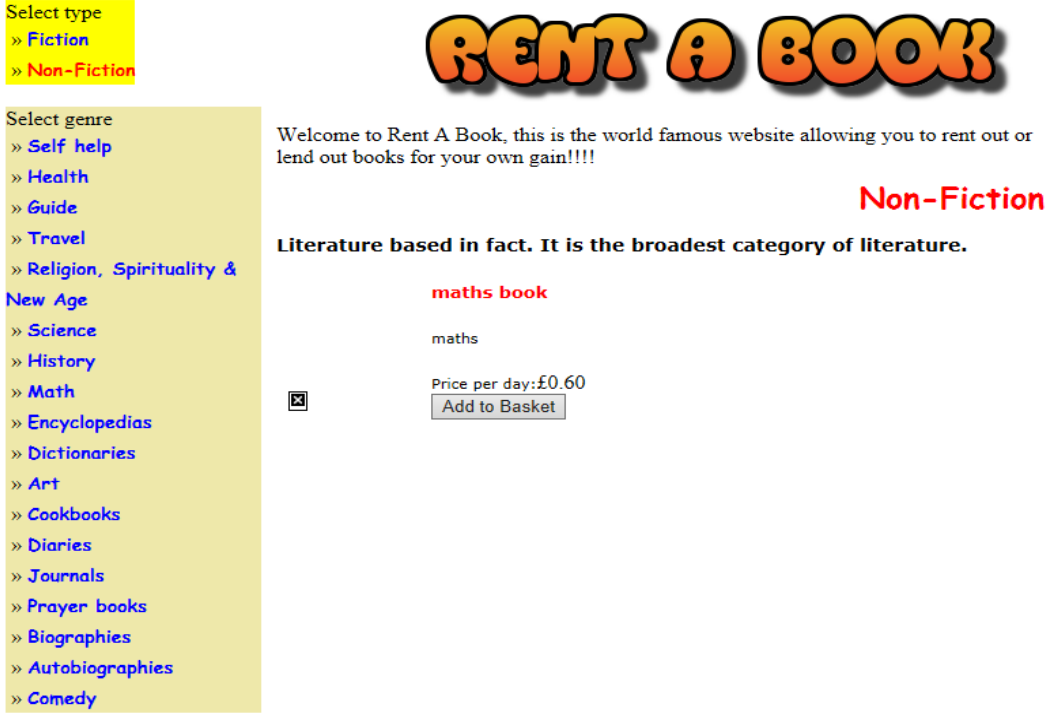


Image 5

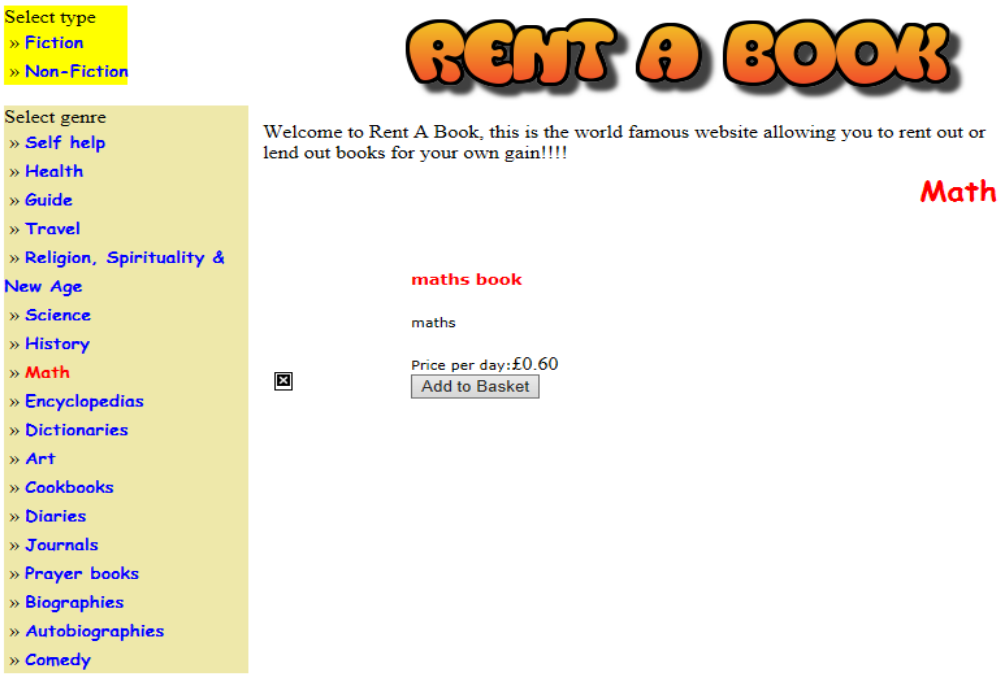


Image 6

Image 7

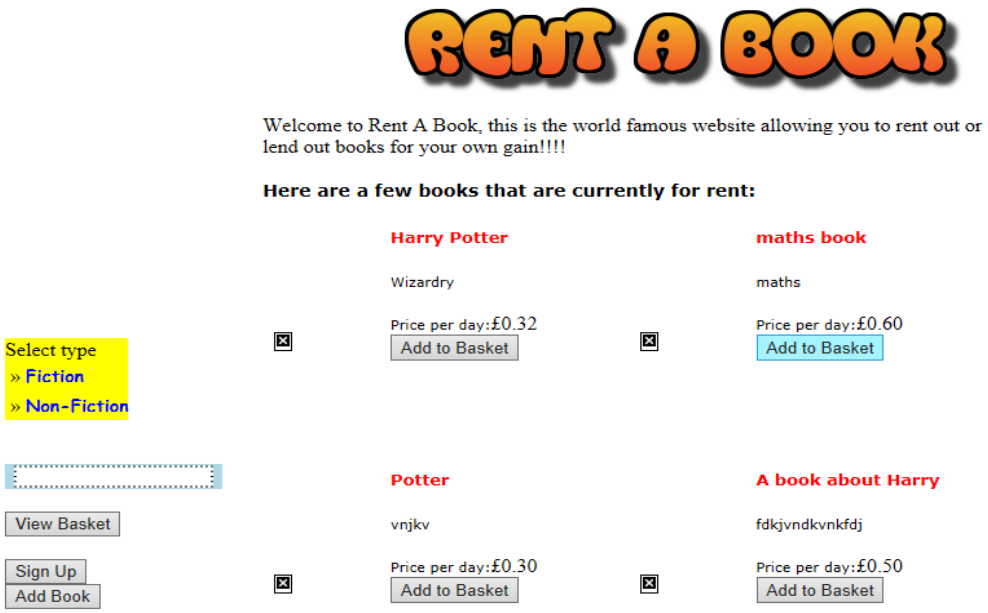


Image 8

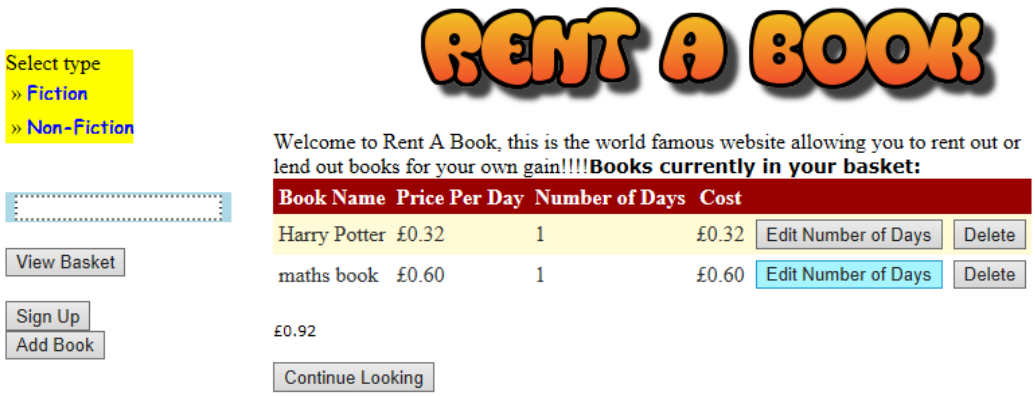


Image 9

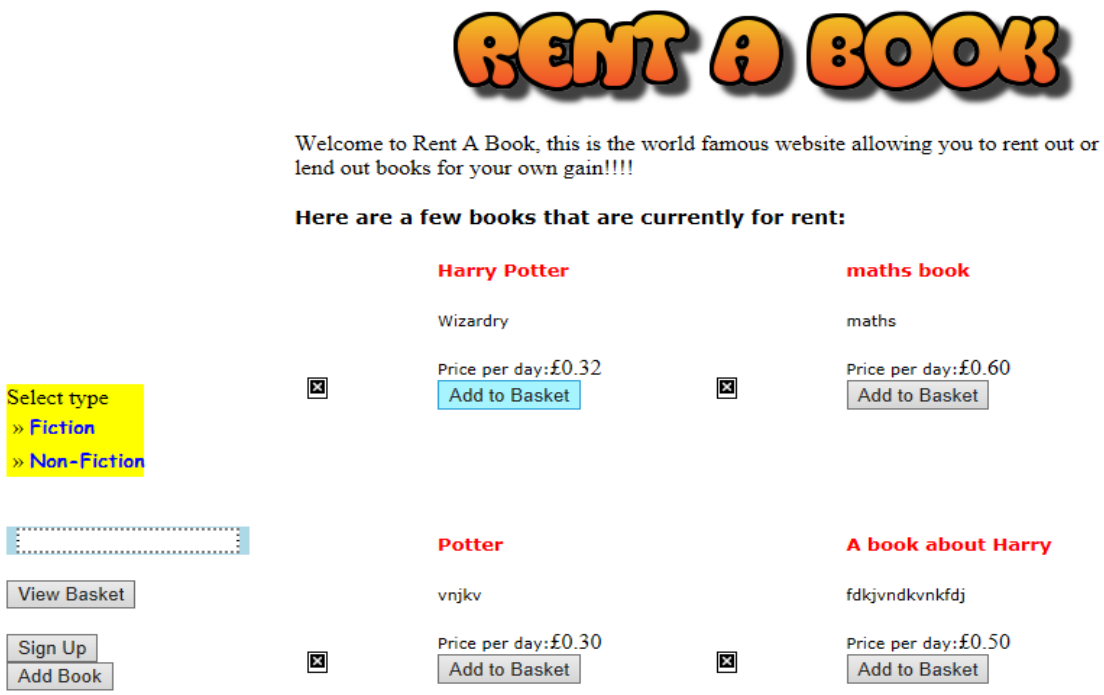


Image 10

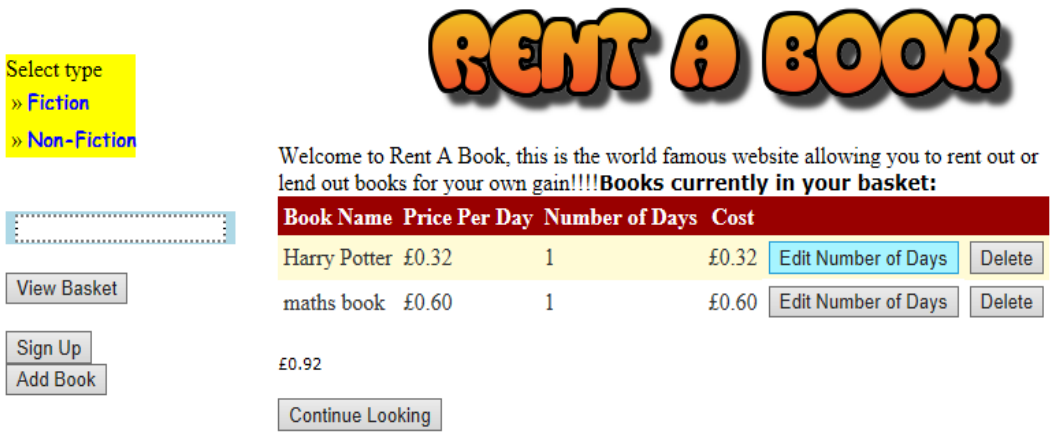


Image 11

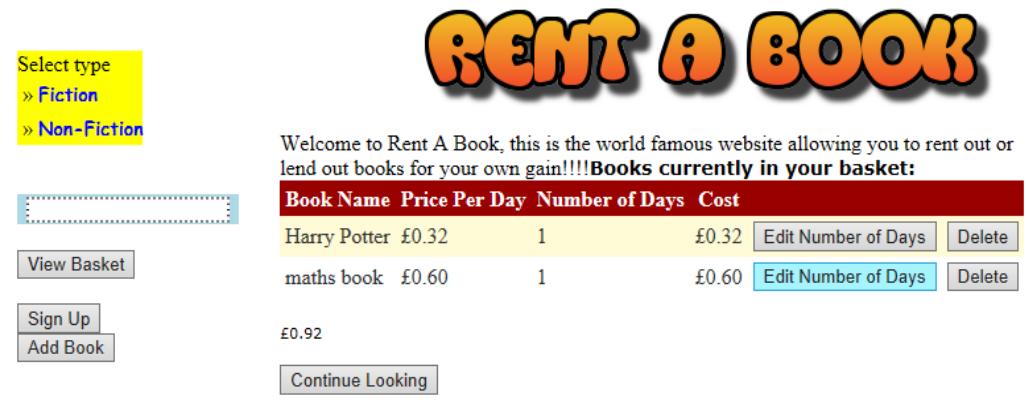


Image 12

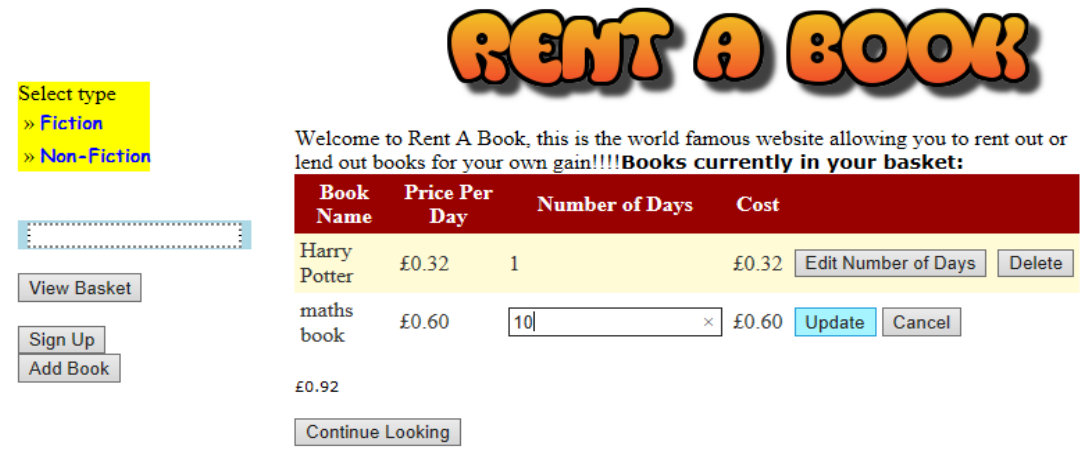


Image 13

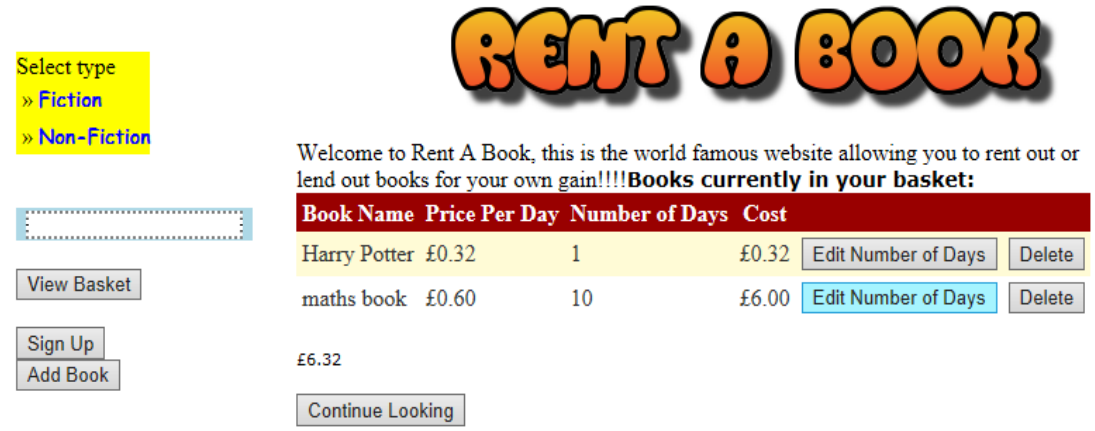


Image 14

Image 15

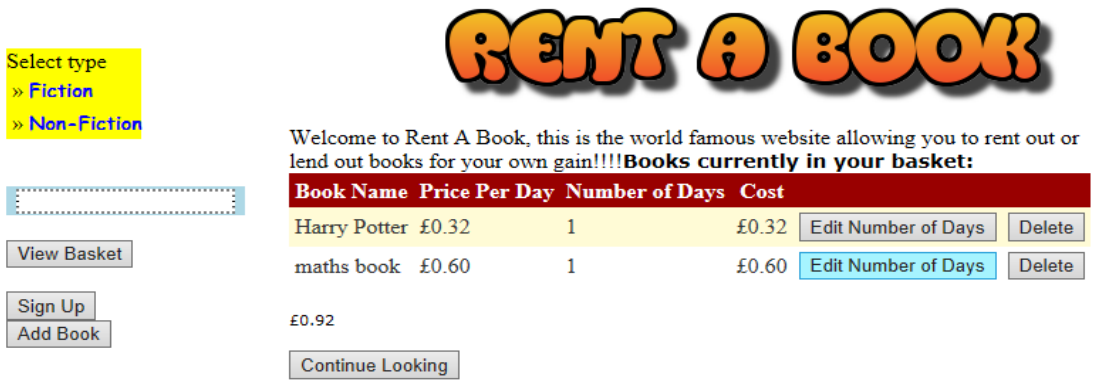


Image 16

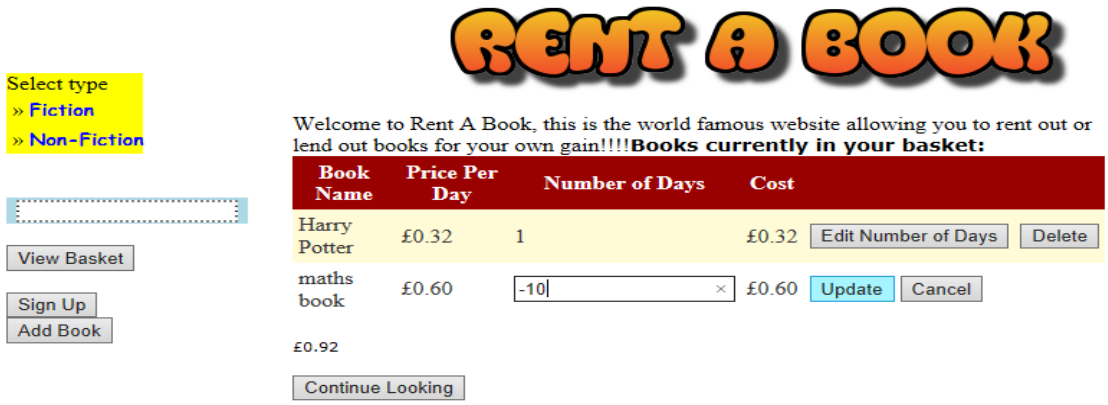


Image 17

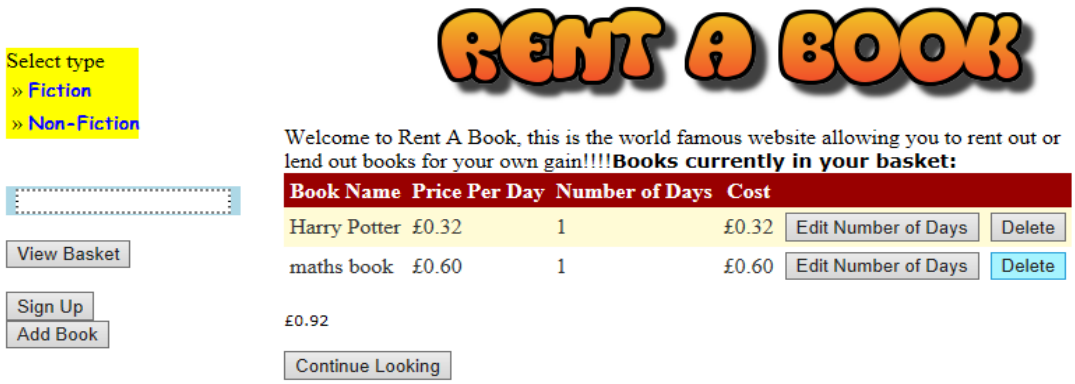


Image 18

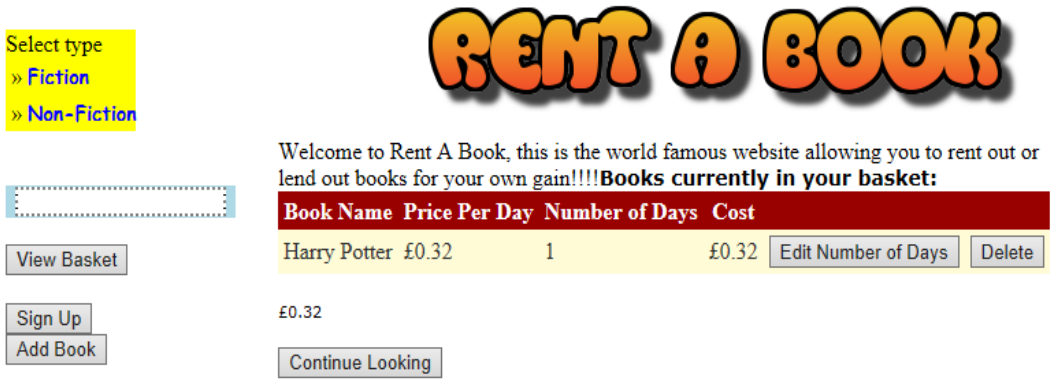


Image 19

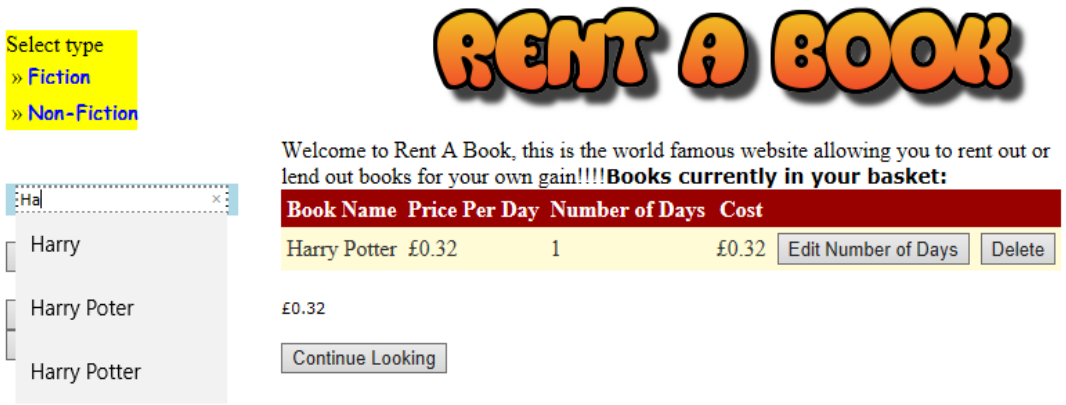


Image 20

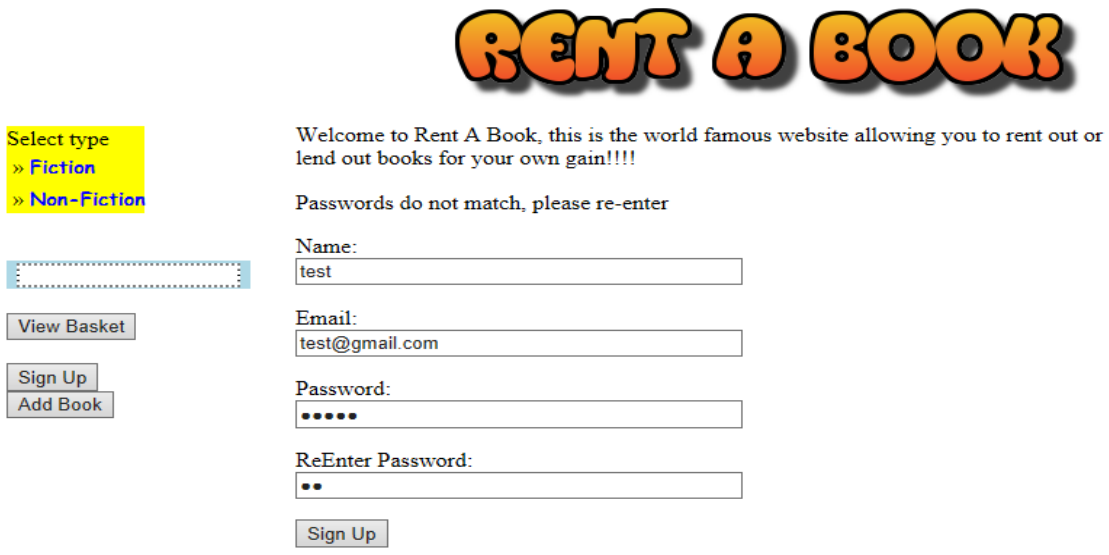


Image 21

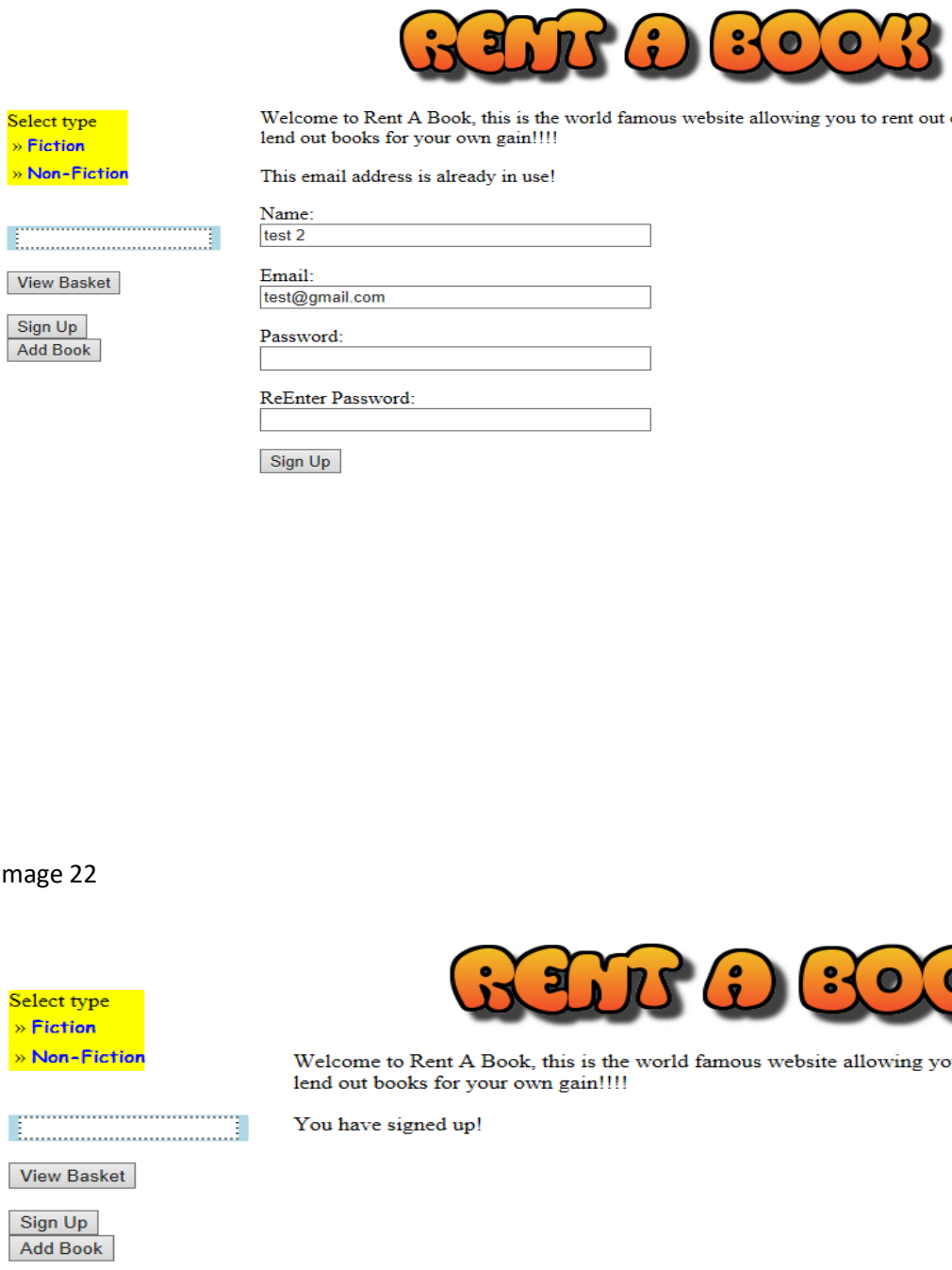


Image 22

Image 23

Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Your Email or Password is incorrect!

Enter Email address:

test@gmail.com

Enter Password:

Name of Book:

test

Description of the Book:

test

Image 24

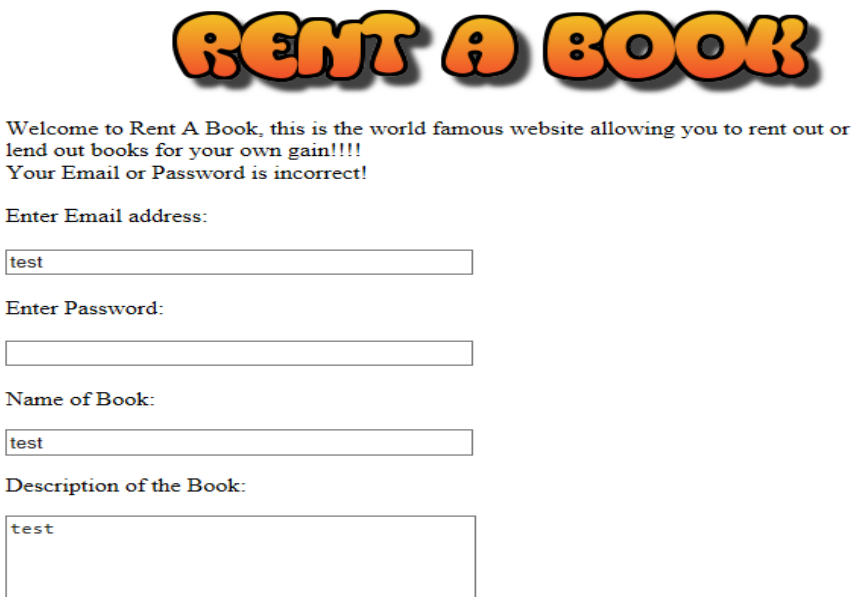


Image 25

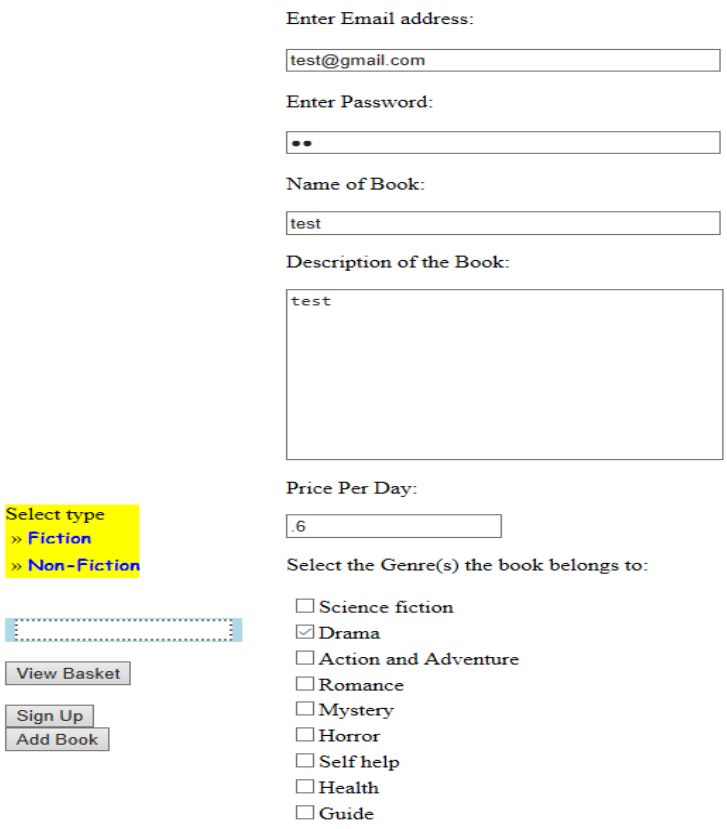


Image 26

Image 27

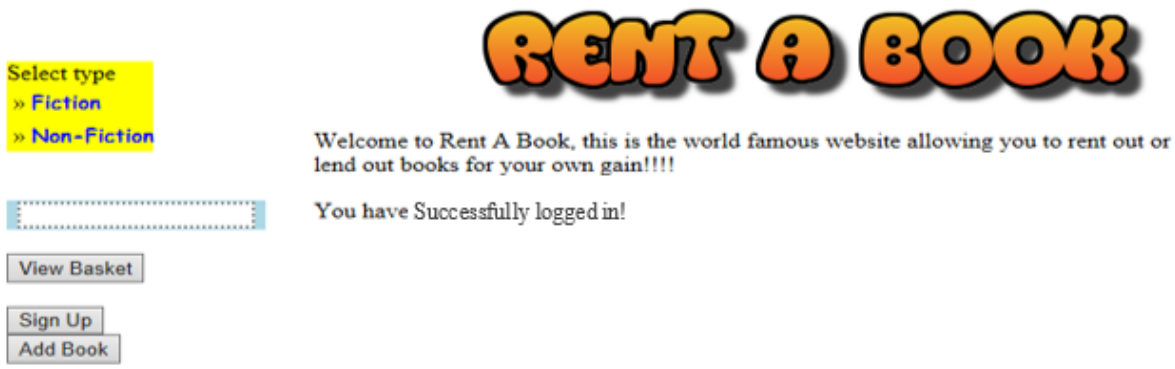


Image 28

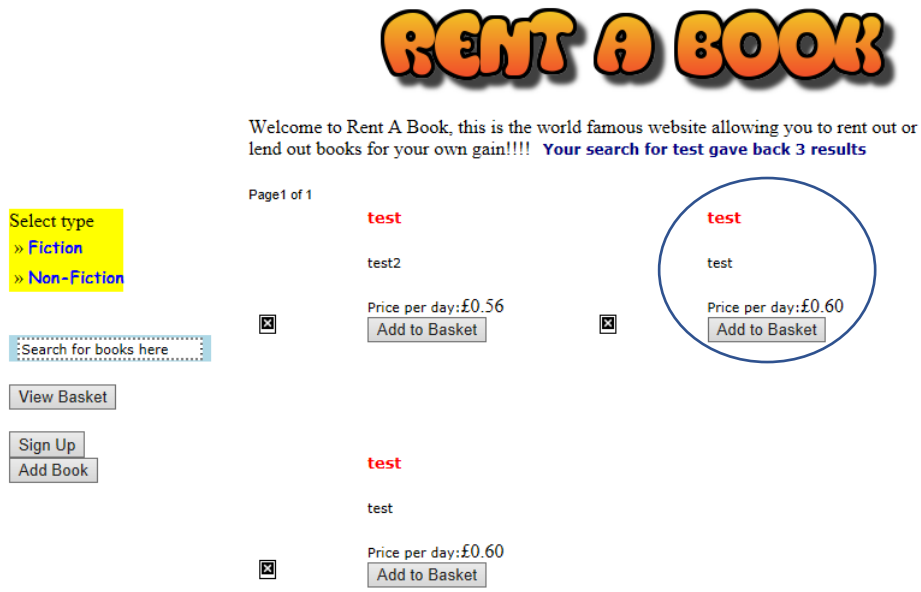


Image 29

Image 30

Image 31

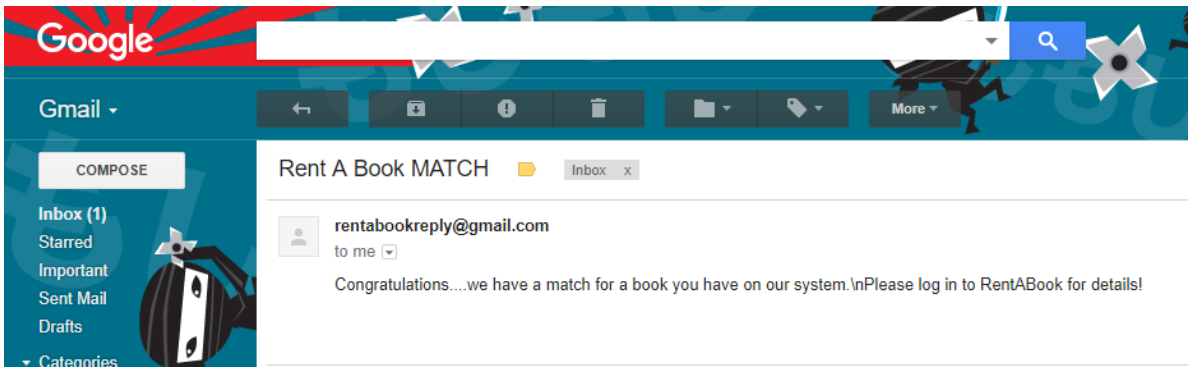


Image 32

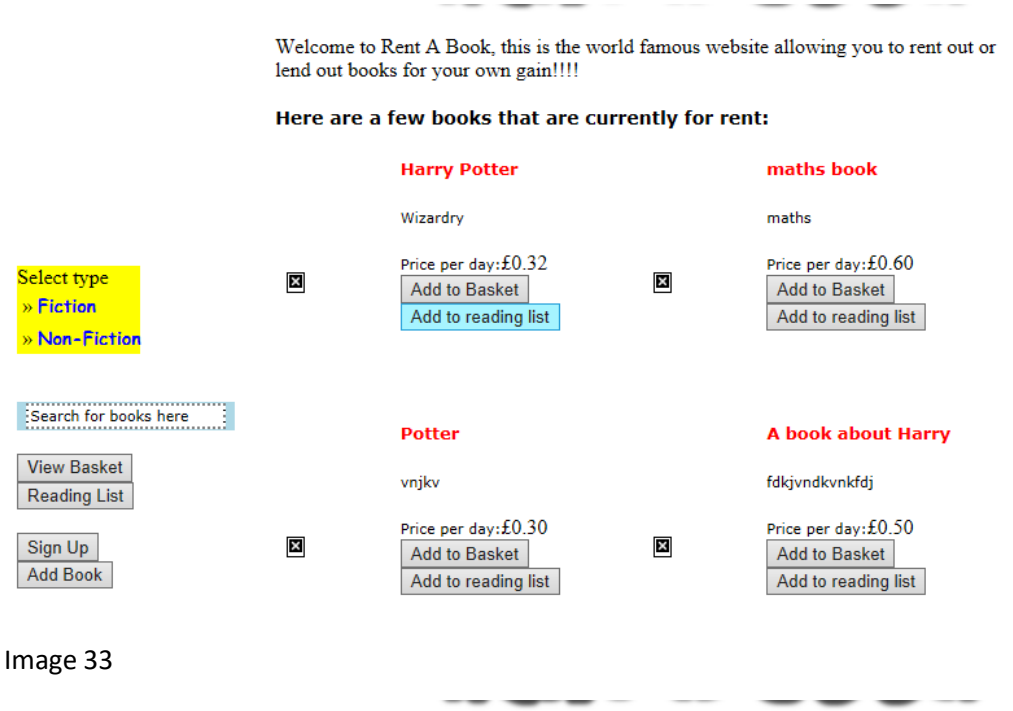
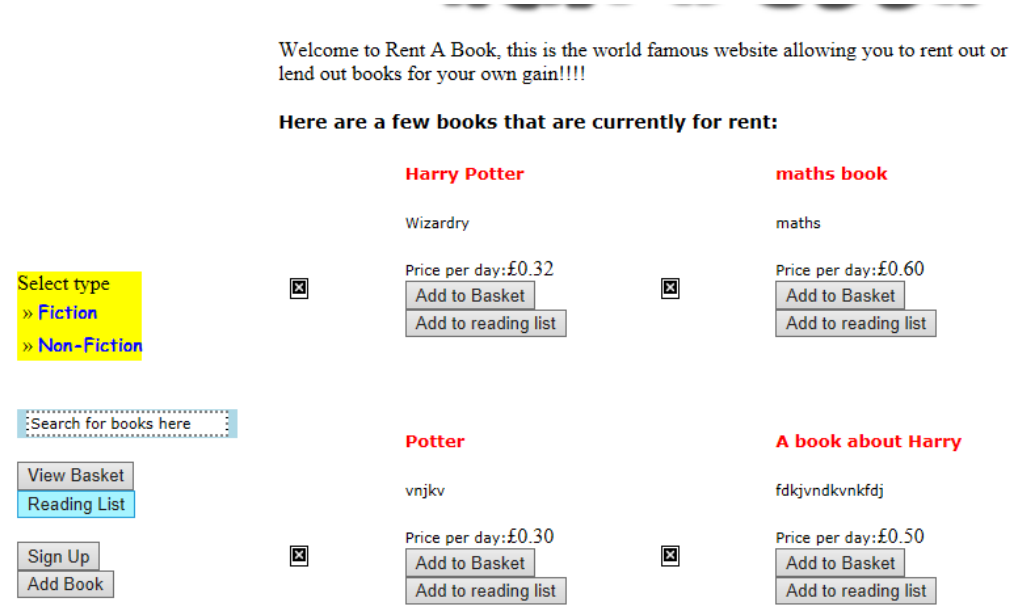

Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Here are a few books that are currently for rent:
Harry Potter
Wizardry
Price per day:£0.32
Add to Basket
Add to reading list
maths book
maths
Price per day:£0.60
Add to Basket
Add to reading list
Select type
» Fiction
» Non-Fiction
Search for books here
View Basket
Reading List
Sign Up
Add Book
Potter
vnjkv
Price per day:£0.30
Add to Basket
Add to reading list
A book about Harry
fdkjvndkvnkfdj
Price per day:£0.50
Add to Basket
Add to reading list

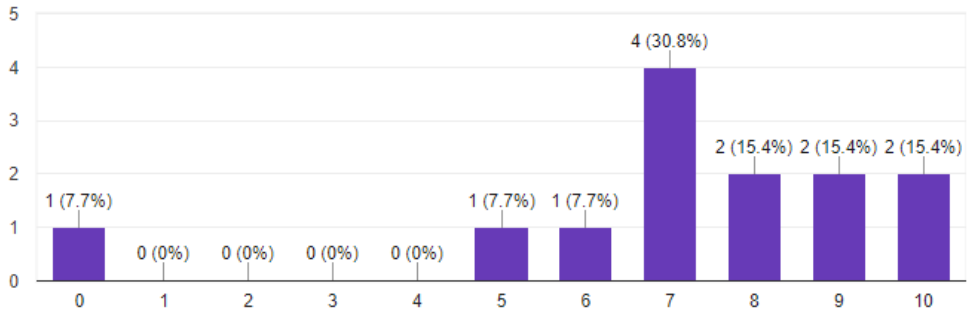
Image 33


Welcome to Rent A Book, this is the world famous website allowing you to rent out or lend out books for your own gain!!!!
Here are a few books that are currently for rent:
Harry Potter
Wizardry
Price per day:£0.32
Add to Basket
Add to reading list
maths book
maths
Price per day:£0.60
Add to Basket
Add to reading list
Select type
» Fiction
» Non-Fiction
Search for books here
View Basket
Reading List
Sign Up
Add Book
Potter
vnjkv
Price per day:£0.30
Add to Basket
Add to reading list
A book about Harry
fdkjvndkvnkfdj
Price per day:£0.50
Add to Basket
Add to reading list

Usability Testing

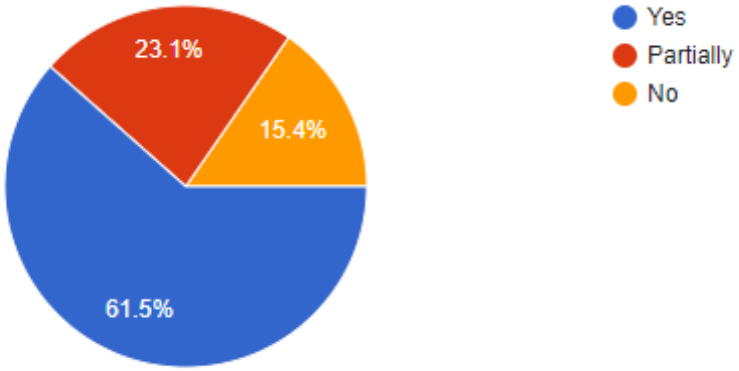
To perform the usability testing I made a user feedback questionnaire for my stakeholders to fill out after using the website. This questionnaire will allow me to quickly get feedback from a wide range of users and be able to summarize the results into pie charts and bar charts.

Please rate how much you enjoyed using the website (0=bad, 10=good)



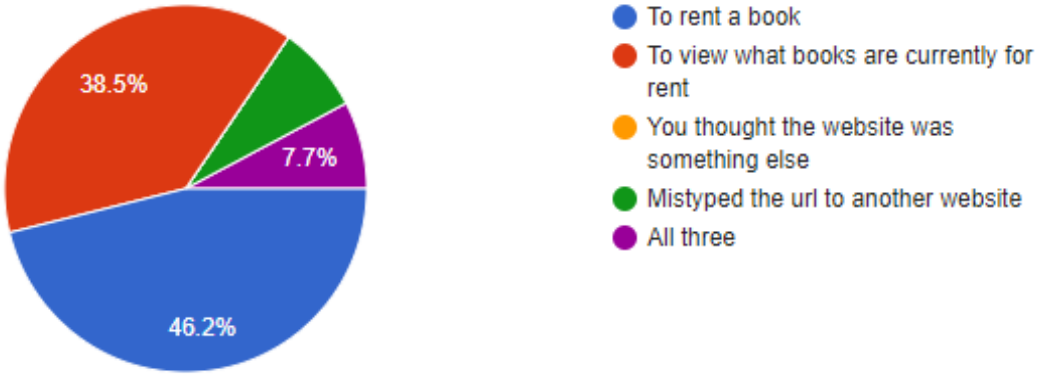
The first question I got my stakeholders to answer was to rate how much they enjoyed using the website. The results showed me that 76.9% of my stakeholders said that they enjoyed using the website and rated 7 or more. This means that most of my stakeholders find the website easy to use and effective, otherwise they would have found the website frustrating/ not enjoyable to use, and instead rate it low.

Did you achieve your goal?



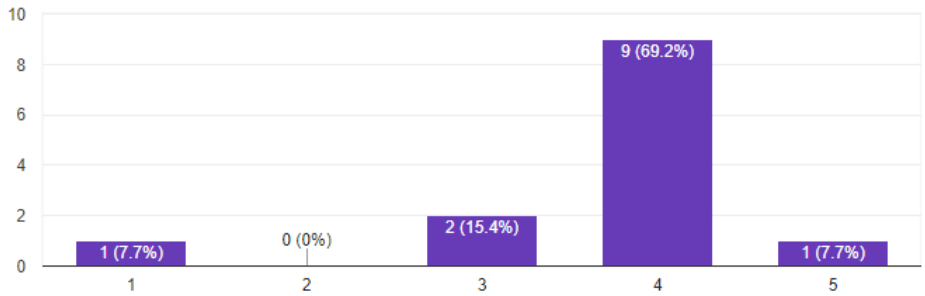
The second question in my questionnaire was if they achieve their goal for entering the website. 61.5% of my stakeholders answered Yes to this question meaning that they left the website happy as they could complete what they came to the website to do. However, 38.5% of the stakeholders either said Partially or No, this shows that my website must be missing features that nearly half of my stakeholders wanted or needed.

What was the reason for your visit?



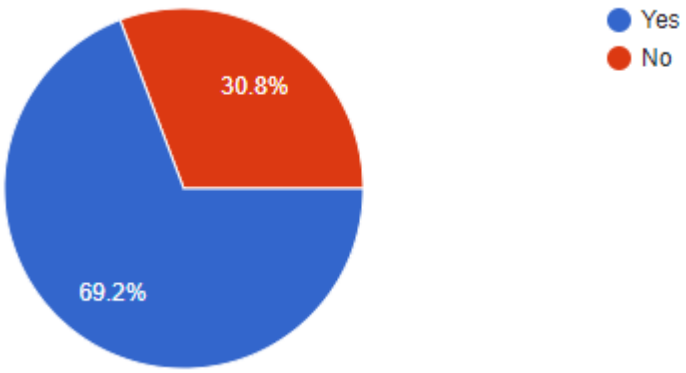
My third question was “What was the reason for you visit?” I asked this question so that I could find out what most of its visitors are doing. For example, if most of the visitors visit to view what books are currently for rent, which is 38.5% of stakeholders, then developing on how these books are presented may be of more use compared to trying to increase the speed of an email being sent to the consumer.

How clear was the website (1=Bad, 5=Good)?



The next question I asked was how clear the website was. A massive majority of my stakeholders, 69.2% of them, said it was 4 out of 5. This shows that most of them found the website easy to use and simple, in terms of renting out a book. However, the results do show that there could be some improvements to make it even more clearer.

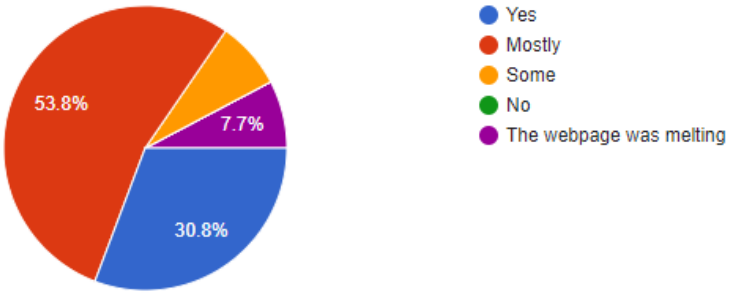
Were you able to successfully rent a book?



My fifth question I had in the questionnaire was “Were you able to successfully rent a book” 69.2% of the stakeholders said yes, however 30.8% of them said no. This means there is either an error in the program somewhere or it is not clear enough for the stakeholders to follow what they need to do.

When I asked the 30.8% of stakeholders who said no, some of them said that they were unable to select or find the book they wanted, while another two said that they typed in their old email or misspelt it when signing up. This shows that I may need to consider adding a feature that will send an email verification to activate their account.

Was the website easy to navigate around?



This question, Was the website easy to navigate around? allowed me to find out how easy the stakeholders found it to go around the website. A large majority, 53.8%, of the stakeholders said Mostly, while another large number of stakeholders said Yes. this tells me that the webpage is very easy to get around and does not take up too much time when searching it.

Were there any parts of the website which were misleading, if so, why was it?

This question was a follow on from the one before it, so that I could get a general idea which parts of the website needs to become more clearer and easier to navigate through. Some of the stakeholders were confused as to where they needed to login as there is no separate button/ part that you need to go through to log in. Instead you only log in when it comes to adding a new book and renting a

book. Some other stakeholders thought that they could sign up just while they are checking out. This shows me that I might need to tell new customers that they need to sign up before trying to rent out books.

Have you ever encountered an error during use? If so what was it

A majority of the stakeholder found no errors during use, however three of them found some. The errors they found was that a few of the pictures of books were unable to be viewed and that they were unable to search for a book called “Oz” This was because when making the solution I set it to search for words more than two letters long, as if I hadn’t nearly every book would be shown due to having words like “a” a couple of times in its title and description. Another error was that one person found that they were charged triple what the rent price was.

Feedback and suggestions for improvement

All my stakeholders said that they liked the website and that they would come back again to the website to rent a book. Many of their suggestions were to improve the design and layout of the website, while others went onto say that it would be better if the website was more interactive. Some of the stakeholders said that the website was very useful and easy to use, however they would have preferred if the money transaction was done through the website or to have been done using PayPal. They also suggested having bigger and bolder text so that it is easier for people with bad eyesight to use the website. Another suggestion was to have books suggested to specific users and to have a separate login pages so that there can be multiple baskets on the one computer.

Evaluation

Evaluate Success Criteria

My first success criteria were that the website must somehow allow the customer to rent a book out. This criterion is essential to my solution as this is the absolute minimum the solution should be able to do, as everything else are features that can be added onto it. My solution has fully met this criteria as the website is able to allow the user to select a book to be added to their basket, as shown in test number 7, from here they are able to get the contact details of the owner to arrange payment and to receive the book for the rented period, as shown in test ().

However, I failed to meet my second success criteria, which was to allow payment through my website to be also done using PayPal. This criterion was only essential if it is the only way the two customers can rent from one another, which it isn't. I was unable to meet this criterion as I was restricted in the amount of time I had to try and connect PayPal to my website. This was because some features such as the search bar and the baskets took longer to make than I anticipated. Although I could create a much simpler way to allow the transaction between the two customers. It was to alert the owner of the book and give them the relevant information for them to get in contact with the person wanting to rent. This allows them to sort out how the payment and the giving of the book is done between themselves. Enabling me to still meet my first success criteria that was to allow the customer to rent out a book.

Another one of my success criteria was being able to search the website for a type of book by selecting and searching its genre. I could fully meet this criteria by allowing the table holding the books to be queried first down to either fiction or non-fiction, showing the results to the user. It is then able to be queried down further as it displays a list of genres underneath for the user to search by, as shown in test numbers 2 to 5.

I was unable to complete the next two success criteria's as I ran out of time and they would be extremely time consuming to try and implement into my program. The two criteria were to implement the predictive and the suggestive search features. I ran out of time to add these features as I underestimated the amount of time it took me to program and fix errors that occurs. These features would have been handy for the user, but they are not essential for the solution to be a success.

The next three successes criteria are essential for the success of the solution as they are about the signup and login pages, and how they need to be able to catch errors and save/read to/from the database. My solution could successfully fully meet all three of them. This is because my website could load the signup page, via the signup button, and is able to check that the details entered are

correct and is not already in the database, as shown in test numbers 18 to 21. The user is then able to log into the correct account when adding a book, as shown in test numbers 22 to 24.

Evaluate Usability Features

The webpage was designed to have a familiar look and feel to other websites. You are initially brought to the main page which is clear and not cluttered, like the google homepage which is massively used by my customers. This allows the customer to immediately start searching for books and has made the whole process more efficient, due to that the users would be slightly familiar to the layout which also makes it easier for them to learn. When adding the books to the basket, they can use the add to basket feature which is very familiar for anybody who is used to using the internet. Within the basket, the customer can easily change the number of days, they wish to rent the book out for, by typing it into the textbox. To remove a book from the basket, the customer only needs to click on one button to perform this action. Also, the look and feel of the basket is like most online shopping baskets, making it easier for the customers to use.

On the homepage the side menu is very clear to the customers and it can be accessed throughout the website, making navigation much easier. In my survey 69.2% of my stakeholders said that they found the website clear and easy to use. However, there is room for improvement as some of stakeholders did not find it easy to navigate around. When the customers are signing up, the textboxes where the customer needs to type in their relevant details are clearly labeled and if they have inputted something incorrectly, they are easily notified of this. Also, when adding a book to the website the owner does not need to type in everything, as there are check and dropdown boxes to make it easier and faster to add them. Not only that, the use of dropdown boxes also makes it more error tolerant as it won't give them the chance to make one. There is even multiple validation checkpoints to stop the user entering in invalid data to the database, and it can inform them of this too. Another usability feature is that whenever the customer starts typing into the search bar, the search bar will show what they previously searched for.

I could improve the checkout area if the project was to continue, as at the minute the owner and customer must arrange the payment transfer themselves. This was because I ran out of time and I was spending time fixing errors. I could improve it so that the website will handle the transaction of money and it could implement PayPal as that is what many of the stakeholders wanted. In the questionnaire the stakeholders also said that I should improve the design of the website and make it more interactive. This would bring in more customers and it would especially attract younger people too.

Limitations & Maintenance

There are a few limitations my solution has. One of the limitations is that the webpage will not function correctly when a customer tries to run the website using a touch interface or on their phone. This is because my aim was to have it can run on a laptop, as this was the most commonly used device by my stakeholder (discovered this during my survey with the stakeholders), which is why I did not focus on making the website compatible with a smartphone or any other small portable device. However, for future maintenance this can be fixed by either making the website compatible, or even better, allow an application version of the website, for the customers to download onto their smartphones and use on the go. This would allow the user to upload straight from their phone, which would most likely have a built-in camera, rather than needing to upload the photo on to the laptop and then onto the website.

Another limitation my current solution has is that it is unable to allow money transactions through it or be able to allow PayPal to be used by the customers. I was unable to add these features onto my current solution due to the limited time and the many business aspects that would be needed, such as opening a bank account to hold the money that is being transferred. Also, the solution would need tight security as it would have to hold card details that hackers would want to get at and steal.

In the future, there could be a lot more customers using the website and they would be creating accounts with it. This means that there will be more user accounts and books saved on the database, meaning it will take up massive amounts of space. To fix this in the future I may need to swap to a bigger server and have backup servers in case the first one goes down. Increasing the power of the server could also be done in the future making it more capable with dealing with large data transaction and changes to the database.

Also because of the increased number of books in the database, the current search algorithm, linear search, will become inefficient as it will take way too long to get the search results. To solve this problem in the future I would need to implement a more efficient search algorithm, such as a binary search for example, and I probably need to keep the data held sorted, so I'll also need to implement a sorting algorithm as well. Another maintenance technique to solve this problem, is to create more tables specific for the different types of genres. This will decrease search time as the algorithm will not need to search through a massive table full of books from different genres, nut instead just go through a smaller table specific to that genre.

A further limitation the current website has is that there is currently no email verification being sent out. This means that the user could type in a valid email with the @ and the .com/.org etc. but they might have typed in an additional letter or left one out, meaning the website will not be able to get in contact with the user. In order to fix this a function called verificationEmail could be made. This function would send a confirmation email to the users typed in email for the user to activate their account. Until then their information could be temporarily saved to a table while awaiting verification, if no verification is returned after a set amount of time the users details can be deleted, else it can be saved to the permanent table.

Output caching could also be enabled to parts of my solution. This will increase the performance of my website as parts of the solution will be able to be loaded from the cache instead of from the server. Doing this will increase the website's load time and saves server processing power.

Another maintenance feature that can be added in the future is more search options, such as being able to select more than one genre when looking for a book. An advanced search could also be added to allow the users to search by the amount of days of the price of the book.

Overall summary

Overall, I believe that the website is an overall success. It can solve the problem my stakeholder had which was to reduce the number of books being thrown away into landfill when only read once, and to give a new life to the books which would have had otherwise been placed in a corner gathering dust and forgotten about.

I have met some challenges along the way, such as fixing the numerous errors when coding my solution, most of which were referencing the wrong variable and syntax errors. Another challenge I faced was that I had limited experience in using the visual basic language and using the .NET framework. However, I overcome these challenges by taking tutorials and developing some test programs. By doing this I gained a better understanding of the framework and libraries.

I am happy with the final solution as it is able to meet all of the briefs provided by the stakeholder. It is also able to be easily learnt as it follows similar standards that are commonly used by other major websites, such as Amazon and eBay. The solution is also very engaging to the user as it has a very user-friendly look and feel to it. The solution is also very effective as it can be easily used by the user to rent out a book and loan out their own books.

Appendix

AddBooksPage.ascx.vb

```
1.  Public Class AddBooksPage
2.      Inherits System.Web.UI.UserControl
3.
4.      Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Ha
    ndles Me.Load
5.
6.      End Sub
7.
8.      Protected Sub AddButton_Click(sender As Object, e As EventArgs) Handles AddButt
    on.Click
9.          Dim Email As String = EmailTextBox.Text
10.         Dim Password As String = PasswordTextBox.Text
11.         Dim BookName As String = NameOfBookTextBox.Text
12.         Dim Description As String = DescriptionOfTheBookTextBox.Text
13.         Dim PricePerDay As String = PriceTextBox.Text
14.         'This will get the list of selected genres
15.         Dim SelectedGenres As ListItem = GenreCheckBoxList.SelectedItem
16.         'This will call the function Checkuser to see if the user entered valid cre
    dentials
17.         Dim authentic As Integer = Account.CheckUser(Email, Password)
18.         'This will check if the variable authentic equals 1
19.         If authentic <> 1 Then
20.             'If it does it means that the user has entered invalid login credential
    s
21.             AddBookLabel.Text = "Your Email or Password is incorrect!"
22.         Else
23.             'Otherwise it will call the following two functions from the Account cl
    ass to save the book to the books table and link the owner of the book to the book

24.             Dim BookId As Integer = Account.AddBooktoTheBooksTable(BookName, Descri
    ption, PricePerDay)
25.             Account.LinkUserAndBook(BookId, Email)
26.         End If
27.
28.
29.     End Sub
30. End Class
```

Basket.ascx.vb

```
1.  Imports System.Net.Mail
2.
3.  Public Class Basket1
4.      Inherits System.Web.UI.UserControl
5.
6.      Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Ha
    ndles MyBase.Load
7.          If Not Page.IsPostBack Then
8.              BindBasket()
9.          End If
10.     End Sub
11.
12.     Private Sub BindBasket()
13.         'This will first check to see the total amount the users basket comes to
14.         Dim cost As Decimal = Basket.GetTotalCost()
15.         TotalCostLabel.Text = String.Format("{0:c}", cost)
16.         'If it comes to 0 then the basket must be empty, this means a message is di
    splayed telling the user
17.         If cost = 0 Then
```

```
18.             IntroductionLabel.Text = "Your basket is empty!!!!"
19.             'This will hide the basket
20.             Grid.Visible = False
21.         Else
22.             'Otherwise it will get the books stored in the basket and display them
23.             Grid.DataSource = Basket.GetBooksFromBasket
24.             Grid.DataBind()
25.         End If
26.     End Sub
27.
28.     Private Sub ContinueLookingButton_Click(sender As Object, e As EventArgs) Handl
    es ContinueLookingButton.Click
29.         Dim redirectPage As String
30.         'This will only run if the user has clicked on the continue looking button
31.         redirectPage = Request.Url.AbsolutePath + "?" + Basket.RemoveBookFromBasket
    ()
32.         Response.Redirect(redirectPage)
33.     End Sub
34.
35.     Private Sub Grid_RowCancelingEdit(sender As Object, e As GridViewCancelEditEven
    tArgs) Handles Grid.RowCancelingEdit
36.         'If the user is on edit mode in the basket and they click cancel this is ca
    lled to take them out of edit mode
37.         Grid.EditIndex = -1
38.         BindBasket()
39.     End Sub
40.
41.     Private Sub Grid_RowDeleting(sender As Object, e As GridViewDeleteEventArgs) Ha
    ndles Grid.RowDeleting
42.         Dim row As GridViewRow = Grid.Rows(e.RowIndex)
43.         Dim bookId As String = Grid.DataKeys(e.RowIndex).Value
44.         'This is run when the user wants to delete a book from their basket
45.         Basket.RemoveFromBasket(bookId)
46.         BindBasket()
47.     End Sub
48.
49.     Private Sub Grid_RowEditing(sender As Object, e As GridViewEditEventArgs) Handl
    es Grid.RowEditing
50.         'This will open the basket in edit mode
51.         Grid.EditIndex = e.NewEditIndex
52.     End Sub
53.
54.     Private Sub Grid_RowUpdating(sender As Object, e As GridViewUpdateEventArgs) Ha
    ndles Grid.RowUpdating
55.         Dim row As GridViewRow = Grid.Rows(e.RowIndex)
56.         Dim bookId As String = Grid.DataKeys(e.RowIndex).Value
57.         Dim days As String = TryCast(row.FindControl("TextBox1"), TextBox).Text
58.         'When the user wants to change the number of days they want to rent a book
    out, this is called
59.         Try
60.             'This will call a function from the basket class to save the new number
     of days
61.             Basket.UpdateBasket(bookId, days)
62.             'However if an exception is thrown, this will catch it and inform the u
    ser they can only enter in a valid number of days
63.         Catch ex As Exception
64.             IntroductionLabel.Text = "You can only enter valid number of days"
65.         Finally
66.             Grid.EditIndex = -1
67.             BindBasket()
68.         End Try
69.     End Sub
70.
```

```
71.     Protected Sub Checkout_Click(sender As Object, e As EventArgs) Handles Checkout
    .Click
72.         'If the user wants to checkout and they clicked the checkout button, this i
    s called
73.         Try
74.             Dim SmtpServer As New SmtpClient
75.             Dim mail As New MailMessage
76.             SmtpServer.UseDefaultCredentials = False
77.             'This will enter in the credentials of the website email account
78.             SmtpServer.Credentials = New Net.NetworkCredential("rentabookreply@gmai
    l.com", "Password")
79.             SmtpServer.Port = 587
80.             SmtpServer.EnableSsl = True
81.             SmtpServer.Host = "smtp.gmail.com"
82.             'These next few lines composes the email
83.             mail = New MailMessage()
84.             mail.From = New MailAddress("rentabookreply@gmail.com")
85.             mail.To.Add("UserEmail")
86.             mail.Subject = "Rent A Book MATCH"
87.             mail.IsBodyHtml = False
88.             'This is the message sent to the owner of the book
89.             mail.Body = "Congratulations....we have a match for a book you have on
    our system.\nPlease log in to RentABook for details!"
90.             SmtpServer.Send(mail)
91.             'This create a popup box telling the user that the message has been sen
    t
92.             MsgBox("Confirmation mail sent")
93.         Catch ex As Exception
94.             MsgBox(ex.ToString)
95.         End Try
96.     End Sub
97. End Class
```

BookList.ascx.vb

```
1.  Public Class BooksList
2.      Inherits System.Web.UI.UserControl
3.
4.      Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Hand
    les MyBase.Load
5.          Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
6.          Dim GenreOfBookID As String = Request.QueryString("GenreOfBookID")
7.          Dim searchString As String = Request.QueryString("Search")
8.          'This will check to see if the user has searched for anything
9.          If Not searchString Is Nothing Then
10.             List.DataSource = Session("searchTable")
11.             List.DataBind()
12.             Session.Remove("searchTable")
13.             'This will see if a genre has been selected
14.         ElseIf Not GenreOfBookID Is Nothing Then
15.             'If so it will call the function GetBooksInGenre from the catalog class

16.             List.DataSource = Catalog.GetBooksInGenre(GenreOfBookID)
17.             List.DataBind()
18.             'This will see if a type of book has been selected
19.         ElseIf Not TypeOfBooksID Is Nothing Then
20.             'If so it will call the function GetBooksOnTypeOfBookPromotion from the
     catalog class
21.             List.DataSource = Catalog.GetBooksOnTypeOfBookPromotion(TypeOfBooksID)

22.             List.DataBind()
23.         Else
24.             List.DataSource = Catalog.GetBooksOnGenreOfBookPromotion()
25.             List.DataBind()
```

```
        End If
    End Sub

    Private Sub list_ItemCommand(ByVal source As Object, ByVal ex As System.Web.UI.
WebControls.DataListCommandEventArgs) Handles List.ItemCommand
        Dim bookId As String = ex.CommandArgument
        Basket.AddBooks(bookId)
    End Sub
```

Catalog.vb

```
Imports System.Data.SqlClient

Public Class TypeOfBookDetails
    Public Name As String
    Public Description As String
End Class

Public Class GenreOfBooksDetails
    Public Name As String
    Public Description As String
End Class

Public Class Catalog
    Public Shared Function GetBookTypes() As SqlDataReader
        'This will open a connection object
        Dim connection As New SqlConnection(connectionString)
        'The line beneath will call the SQL stored procedure GetTypeOfBooks
        Dim command As New SqlCommand("GetTypeOfBooks", connection)
        command.CommandType = CommandType.StoredProcedure
        connection.Open()
        'This will return an SqlDataReader
        Return command.ExecuteReader(CommandBehavior.CloseConnection)
    End Function

    Private Shared ReadOnly Property connectionString() As String
        Get
            ' Return ConfigurationSettings.AppSettings("ConnectionString")
            Return System.Configuration.ConfigurationManager.AppSettings("Connectio
nString")

        End Get
    End Property

    Public Shared Function GetTypeOfBookDetails(ByVal TypeOfBooksId As String) As T
ypeOfBookDetails
        Dim connection As New SqlConnection(connectionString)
        'The line beneath will call the SQL stored procedure GetTypeOfBooksDetails

        Dim command As New SqlCommand("GetTypeOfBooksDetails", connection)
        command.CommandType = CommandType.StoredProcedure
        'These are the parameters being passed in
        command.Parameters.Add("@TypeOfBooksID", SqlDbType.Int, 4)
        command.Parameters("@TypeOfBooksID").Value = TypeOfBooksId
        command.Parameters.Add("@Name", SqlDbType.VarChar, 50)
        command.Parameters("@Name").Direction = ParameterDirection.Output
        command.Parameters.Add("@Description", SqlDbType.VarChar, 200)
        command.Parameters("@Description").Direction = ParameterDirection.Output
        Try
            connection.Open()
            command.ExecuteNonQuery()
        Finally
            connection.Close()
        End Try
```

```
51.         Dim details As New TypeOfBookDetails()
52.         details.Name = command.Parameters("@Name").Value.ToString()
53.         details.Description = command.Parameters("@Description").Value.ToString()
54.         Return details
55.     End Function
56.
57.     Public Shared Function GetGenresInTypeOfBook(ByVal TypeOfBooksId As String) As
    SqlDataReader
58.         Dim connection As New SqlConnection(connectionString)
59.         'The line beneath will call the SQL stored procedure GetGenreFromTypeOfBook
    s
60.         Dim command As New SqlCommand("GetGenreFromTypeOfBooks", connection)
61.         command.CommandType = CommandType.StoredProcedure
62.         'This is the the parameter being passed in
63.         command.Parameters.Add("@TypeOfBooksID", SqlDbType.Int, 4)
64.         command.Parameters("@TypeOfBooksID").Value = TypeOfBooksId
65.         Try
66.             connection.Open()
67.             Return command.ExecuteReader(CommandBehavior.CloseConnection)
68.         Catch ex As Exception
69.             connection.Close()
70.             Throw ex
71.         End Try
72.     End Function
73.
74.     Public Shared Function GetGenreOfBookDetails(ByVal GenreOfBooksId As String) As
     GenreOfBooksDetails
75.         Dim connection As New SqlConnection(connectionString)
76.         'The line beneath will call the SQL stored procedure GetgenreOfBookDetails

77.         Dim command As New SqlCommand("GetGenreOfBookDetails", connection)
78.         command.CommandType = CommandType.StoredProcedure
79.         'These are the parameters being passed through
80.         command.Parameters.Add("@GenreOfBookID", SqlDbType.Int, 4)
81.         command.Parameters("@GenreOfBookID").Value = GenreOfBooksId
82.         command.Parameters.Add("@Name", SqlDbType.VarChar, 50)
83.         command.Parameters("@Name").Direction = ParameterDirection.Output
84.         command.Parameters.Add("@Description", SqlDbType.VarChar, 200)
85.         command.Parameters("@Description").Direction = ParameterDirection.Output
86.         Try
87.             connection.Open()
88.             command.ExecuteNonQuery()
89.         Finally
90.             connection.Close()
91.         End Try
92.         Dim details As New GenreOfBooksDetails()
93.         details.Name = command.Parameters("@Name").Value.ToString()
94.         details.Description = command.Parameters("@Description").Value.ToString()
95.         Return details
96.     End Function
97.
98.     Public Shared Function GetBooksInGenre(ByVal GenreOfBooksId As String) As SqlDa
    taReader
99.         Dim connection As New SqlConnection(connectionString)
100.                'The line beneath will call the SQL stored procedure GetbooksInGenre

101.                Dim command As New SqlCommand("GetBooksInGenre", connection)
102.                command.CommandType = CommandType.StoredProcedure
103.                'This is the parameter being passed in
104.                command.Parameters.Add("@GenreOfBookID", SqlDbType.Int, 4)
105.                command.Parameters("@GenreOfBookID").Value = GenreOfBooksId
106.                Try
107.                    connection.Open()
108.                    Return command.ExecuteReader(CommandBehavior.CloseConnection)
109.                Catch ex As Exception
110.                    connection.Close()
```

```
111.                 Throw ex
112.             End Try
113.         End Function
114.
115.         Public Shared Function GetBooksOnTypeOfBookPromotion(ByVal TypeOfBooksID
     As String) As SqlDataReader
116.             Dim connection As New SqlConnection(connectionString)
117.             Dim command As New SqlCommand("GetBooksOnTypeOfBookPromotion", conne
    ction)
118.             command.CommandType = CommandType.StoredProcedure
119.             'This will pass through a parameter called TypeOfBooksID
120.             command.Parameters.Add("@TypeOfBooksID", SqlDbType.Int, 4)
121.             command.Parameters("@TypeOfBooksID").Value = TypeOfBooksID
122.             Try
123.                 connection.Open()
124.                 Return command.ExecuteReader(CommandBehavior.CloseConnection)
125.             Catch ex As Exception
126.                 connection.Close()
127.                 Throw ex
128.             End Try
129.         End Function
130.
131.         Public Shared Function GetBooksOnGenreOfBookPromotion() As SqlDataReader

132.             Dim connection As New SqlConnection(connectionString)
133.             Dim command As New SqlCommand("GetBooksOnGenrePromotion", connection
    )
134.             command.CommandType = CommandType.StoredProcedure
135.             Try
136.                 connection.Open()
137.                 Return command.ExecuteReader(CommandBehavior.CloseConnection)
138.             Catch ex As Exception
139.                 connection.Close()
140.                 Throw ex
141.             End Try
142.         End Function
143.
144.         Public Shared Function SearchCatalog(ByVal searchString As String, ByVal
     pageNumber As String, ByVal booksOnPage As String) As Integer
145.             Dim connection As New SqlConnection(connectionString)
146.             Dim command As New SqlCommand("SearchCatalog", connection)
147.             command.CommandType = CommandType.StoredProcedure
148.             'This will pass through the parameters @pageNumber, @BooksOnPage and
     @HowManyResults into the SQL stored procedure
149.             command.Parameters.Add("@PageNumber", SqlDbType.Int)
150.             command.Parameters("@PageNumber").Value = pageNumber
151.             command.Parameters.Add("@BooksOnPage", SqlDbType.Int)
152.             command.Parameters("@BooksOnPage").Value = booksOnPage
153.             command.Parameters.Add("@HowManyResults", SqlDbType.Int)
154.             command.Parameters("@HowManyResults").Direction = ParameterDirection
    .Output
155.
156.             Dim words() As String = Split(searchString, " ")
157.             Dim wordCount As Integer = words.Length
158.             Dim index As Integer = 0
159.             Dim addedwords As Integer = 0
160.             'This will keep looping around until eiter addedwords equal 5 or les
    s, or index is smaller than wordCount
161.             While addedwords < 5 And index < wordCount
162.                 'This checks if the length of the word is bigger than two
163.                 If Len(words(index)) > 2 Then
164.                     'This is so that the resuls will not come up with every book
     with a single letter that matches it
165.                     addedwords += 1
166.                     command.Parameters.AddWithValue("@word" + addedwords.ToStrin
    g, words(index))
```

```
                    End If
                    index += 1
                End While

                Try
                    connection.Open()
                    Dim reader As SqlDataReader
                    reader = command.ExecuteReader(CommandBehavior.CloseConnection)

                    Dim table As New DataTable()
                    Dim fieldCount As Integer = reader.FieldCount
                    Dim fieldindex As Integer
                    For fieldindex = 0 To fieldCount - 1
                        table.Columns.Add(reader.GetName(fieldindex), reader.GetFiel
dType(fieldindex))
                    Next
                    Dim row As DataRow
                    While reader.Read()
                        row = table.NewRow()
                        For fieldindex = 0 To fieldCount - 1
                            row(fieldindex) = reader(fieldindex)
                        Next
                        table.Rows.Add(row)
                    End While
                    reader.Close()
                    HttpContext.Current.Session("searchTable") = table
                    Return command.Parameters("@HowManyResults").Value
                Catch ex As Exception
                    connection.Close()
                    Throw ex
                End Try
            End Function
        End Class
```

Catalog.ascx.vb

```
Public Class Catalog1
    Inherits System.Web.UI.UserControl

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Ha
ndles MyBase.Load
        Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
        Dim GenreOfBookID As String = Request.QueryString("GenreOfBookID")
        'This will check if one of the genres have been clicked on
        If Not GenreOfBookID Is Nothing Then
            Dim GenreOfBooksDetails As New GenreOfBooksDetails()
            'If it has then this will set the title of the page too its name and a
description of it is shown too
            'The line below calls a function from the catalog class to get this inf
ormation from the table
            GenreOfBooksDetails = Catalog.GetGenreOfBookDetails(GenreOfBookID)
            sectionTitleLabel.Text = GenreOfBooksDetails.Name
            descriptionLabel.Text = GenreOfBooksDetails.Description
            'If a genre hasent been selected it will then check to see if a type of
 book has been selected
        ElseIf Not TypeOfBooksID Is Nothing Then
            'If it has then this will set the title of the page too its name and a
description of it is shown too
            'The line below calls a function from the catalog class to get this inf
ormation from the table
            Dim TypeOfBookDetails As New TypeOfBookDetails()
            TypeOfBookDetails = Catalog.GetTypeOfBookDetails(TypeOfBooksID)
```

```
            sectionTitleLabel.Text = TypeOfBookDetails.Name
            descriptionLabel.Text = TypeOfBookDetails.Description
        End If
    End Sub

End Class
```

Account.vb

```
Imports System.Data.SqlClient

Public Class Account

    Private Shared ReadOnly Property connectionString() As String
        Get
            ' Return ConfigurationSettings.AppSettings("ConnectionString")
            Return System.Configuration.ConfigurationManager.AppSettings("ConnectionString")

        End Get
    End Property

    Public Shared Function AddUser(ByVal Name As String, ByVal Email As String, ByVal Password As String)
        Dim connection As New SqlConnection(connectionString)
        'This will load the sql procedure AddtoUserTable
        Dim command As New SqlCommand("AddToUserTable", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@name", SqlDbType.Char, 36)
        command.Parameters("@name").Value = Name
        command.Parameters.Add("@email", SqlDbType.Char, 36)
        command.Parameters("@email").Value = Email
        command.Parameters.Add("@password", SqlDbType.Char, 36)
        command.Parameters("@password").Value = Password
        Try
            'This will open the connecetion to the database which holds the procedures
            connection.Open()
            command.ExecuteNonQuery()
        Finally
            connection.Close()
        End Try
        Return 0
    End Function

    Public Shared Function CheckUser(ByVal Email As String, ByVal Password As String)
        Dim connection As New SqlConnection(connectionString)
        'This will load the sql procedure CheckUser
        Dim command As New SqlCommand("CheckUser", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@email", SqlDbType.Char, 36)
        command.Parameters("@email").Value = Email
        command.Parameters.Add("@password", SqlDbType.Char, 36)
        command.Parameters("@password").Value = Password
        command.Parameters.Add("@Emailwordcount", SqlDbType.Int)
        command.Parameters("@Emailwordcount").Direction = ParameterDirection.Output

        Dim authentic As Integer = 0
        Try
            'This will open the connecetion to the database which holds the procedures
            connection.Open()
```

```
            command.ExecuteNonQuery()
            'This will save the results to the variable called authentic
            authentic = command.Parameters("@Emailwordcount").Value
        Finally
            connection.Close()
        End Try
        'This passes out the variable to the calling function
        Return authentic
    End Function

    Public Shared Function AddBooktoTheBooksTable(ByVal Name As String, ByVal Descr
iption As String, ByVal PricePerDay As String)
        Dim connection As New SqlConnection(connectionString)
        'This will load the sql procedure AddtoThe BooksTable
        Dim command As New SqlCommand("AddToTheBooksTable", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@bookName", SqlDbType.Char, 36)
        command.Parameters("@bookName").Value = Name
        command.Parameters.Add("@description", SqlDbType.Char, 36)
        command.Parameters("@description").Value = Description
        command.Parameters.Add("@pricePerDay", SqlDbType.Char, 36)
        command.Parameters("@pricePerDay").Value = Convert.ToDouble(PricePerDay)
        command.Parameters.Add("@bookId", SqlDbType.Int, 36)
        command.Parameters("@bookId").Direction = ParameterDirection.Output
        Dim bookId As Integer = 0
        Try
            'This will open the connecetion to the database which holds the procedu
res
            connection.Open()
            command.ExecuteNonQuery()
            bookId = command.Parameters("@bookId").Value
        Finally
            connection.Close()
        End Try
        Return bookId
    End Function

    Public Shared Function LinkUserAndBook(ByVal BookId As Integer, ByVal Email As
String)
        Dim userID As Integer = getUserId(Email)
        Dim connection As New SqlConnection(connectionString)

        'This will load the sql procedure AddtoUserJunction
        Dim command As New SqlCommand("AddToUserJunction", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add("@bookId", SqlDbType.Int, 36)
        command.Parameters("@bookId").Value = BookId
        command.Parameters.Add("@userId", SqlDbType.Char, 36)
        command.Parameters("@UserId").Value = userID
        Try
            'This will open the connecetion to the database which holds the procedu
res
            connection.Open()
            command.ExecuteNonQuery()
                Finally
                    connection.Close()
                End Try
                Return 0
            End Function

            Private Shared Function getUserId(ByVal Email As String)
                Dim connection As New SqlConnection(connectionString)
                'This will load the sql procedure GetUserId
                Dim command As New SqlCommand("GetUserId", connection)
                command.CommandType = CommandType.StoredProcedure
                command.Parameters.Add("@email", SqlDbType.Char, 36)
```

```
                command.Parameters("@email").Value = Email
                command.Parameters.Add("@userId", SqlDbType.Int, 36)
                command.Parameters("@userId").Direction = ParameterDirection.Output

                Dim userId As Integer = 0
                Try
                    'This will open the connecetion to the database which holds the procedures
                    connection.Open()
                    command.ExecuteNonQuery()
                    userId = command.Parameters("@userId").Value
                Finally
                    connection.Close()
                End Try
                Return userId

            End Function
        End Class
```

DepartmentList.ascx.vb

```
Public Class DepartmentsList
    Inherits System.Web.UI.UserControl

    Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        'The departmentIndex parameter is added to the query string when a department link is clicked.
        'This is because that when the page is reloaded, the DataList forgets which link was clicked.
        Dim listIndex As String = Request.QueryString("departmentIndex")
        'If listIndex has a value, the user has clicked on a type of book
        If Not listIndex Is Nothing Then
            List.SelectedIndex = CInt(listIndex)
        End If
        'Get departments returns a SqlDataReader object that has two fields: TypeOfBookID and Name.
        'These fields are read in the SelectedItemTemplate and ItemTemplate of the DataList
        List.DataSource = Catalog.GetBookTypes()
        List.DataBind()
    End Sub

End Class
```

GenreList.ascx.vb

```
Public Class WebUserControl1
    Inherits System.Web.UI.UserControl

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
        'This will check if a type of book has been clicked
        If Not TypeOfBooksID Is Nothing Then
            Dim listIndex As String = Request.QueryString("GenreIndex")
            If Not listIndex Is Nothing Then
                list.SelectedIndex = CInt(listIndex)
            End If
            'If one of them have been clicked, it will display the list of genres belonging to that type of book
```

```
            list.DataSource = Catalog.GetGenresInTypeOfBook(TypeOfBooksID)
            list.DataBind()
        End If
    End Sub

End Class
```

ReadList.ascx.vb

```
Public Class ReadList
    Inherits System.Web.UI.UserControl

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

    End Sub

    Protected Sub ContinueLookingButton_Click(sender As Object, e As EventArgs) Handles ContinueLookingButton.Click
        'This will run only if the user has clicked the continue looking button
        Dim redirectPage As String
        redirectPage = Request.Url.AbsolutePath + "?" + Basket.RemoveBookFromBasket()
        Response.Redirect(redirectPage)
    End Sub

    Protected Sub Grid_SelectedIndexChanged(sender As Object, e As EventArgs) Handles Grid.SelectedIndexChanged

    End Sub


    Private Sub BindBasket()
        'This will get the books for the read list grid
        Grid.DataSource = ReadList.GetBooksFromReadList
        Grid.DataBind()
    End Sub

    Private Sub Grid_RowDeleting(sender As Object, e As GridViewDeleteEventArgs) Handles Grid.RowDeleting
        'This will run if the user wants to delete a book from their read list
        Dim row As GridViewRow = Grid.Rows(e.RowIndex)
        Dim bookId As String = Grid.DataKeys(e.RowIndex).Value
        Basket.RemoveFromBasket(bookId)
        BindBasket()
    End Sub
End Class
```

Basket.vb

```
Imports System.Data.SqlClient

Public Class Basket
    Friend Shared EditItemIndex As Integer

    Private Shared ReadOnly Property basketId()
        Get
            Dim context As HttpContext = HttpContext.Current
            'This will check if the user has cookies
            If context.Request.Cookies("RentABook_BasketID") Is Nothing Then
                'If they do not it will create one
                Dim cartID As Guid = Guid.NewGuid()
```

```
13.                 Dim cookie As New HttpCookie("RentABook_BasketID", cartID.ToString)
14.                 Dim currentDate As DateTime = DateTime.Now()
15.                 Dim expiry As DateTime = currentDate.Add(New TimeSpan(10, 0, 0, 0))
16.                 cookie.Expires = expiry
17.                 context.Response.Cookies.Add(cookie)
18.                 Return cartID.ToString
19.             Else
20.                 Return context.Request.Cookies("RentABook_BasketID").Value
21.             End If
22.         End Get
23.     End Property
24.
25.     Private Shared ReadOnly Property connectionString() As String
26.         Get
27.             Return ConfigurationManager.AppSettings("connectionString")
28.         End Get
29.     End Property
30.
31.     Public Shared Function AddBooks(ByVal bookId As String)
32.         Dim connection As New SqlConnection(connectionString)
33.         'This will load the sql procedure AddtoBasket
34.         Dim command As New SqlCommand("AddToBasket", connection)
35.         command.CommandType = CommandType.StoredProcedure
36.         command.Parameters.Add("@basketID", SqlDbType.Char, 36)
37.         command.Parameters("@basketID").Value = basketId
38.         command.Parameters.Add("@BookID", SqlDbType.Int)
39.         command.Parameters("@BookID").Value = bookId
40.         Try
41.             'This will open the connecetion to the database which holds the procedu
    res
42.             connection.Open()
43.             command.ExecuteNonQuery()
44.         Finally
45.             connection.Close()
46.         End Try
47.         Return 0
48.     End Function
49.
50.     Public Shared Function UpdateBasket(ByVal bookId As String, ByVal days As Integ
    er)
51.         Dim connection As New SqlConnection(connectionString)
52.         'This will load the sql procedure UpdateBasket
53.         Dim command As New SqlCommand("UpdateBasket", connection)
54.         command.CommandType = CommandType.StoredProcedure
55.         command.Parameters.Add("@basketID", SqlDbType.Char, 36)
56.         command.Parameters("@basketID").Value = basketId
57.         command.Parameters.Add("@BookID", SqlDbType.Int)
58.         command.Parameters("@BookID").Value = bookId
59.         command.Parameters.Add("@Days", SqlDbType.Int)
60.         command.Parameters("@Days").Value = days
61.         Try
62.             'This will open the connecetion to the database which holds the procedu
    res
63.             connection.Open()
64.             command.ExecuteNonQuery()
65.         Finally
66.             connection.Close()
67.         End Try
68.         Return 0
69.     End Function
70.
71.     Public Shared Function RemoveFromBasket(ByVal bookId As String)
72.         Dim connection As New SqlConnection(connectionString)
73.         'This will load the sql procedure RemoveFromBasket
```

```
74.         Dim command As New SqlCommand("RemoveFromBasket", connection)
75.         command.CommandType = CommandType.StoredProcedure
76.         command.Parameters.Add("@basketID", SqlDbType.Char, 36)
77.         command.Parameters("@basketID").Value = basketId
78.         command.Parameters.Add("@BookID", SqlDbType.Int)
79.         command.Parameters("@BookID").Value = bookId
80.         Try
81.             'This will open the connecetion to the database which holds the procedu
    res
82.             connection.Open()
83.             command.ExecuteNonQuery()
84.         Finally
85.             connection.Close()
86.         End Try
87.         Return 0
88.     End Function
89.
90.     Public Shared Function GetBooksFromBasket() As SqlDataReader
91.         Dim connection As New SqlConnection(connectionString)
92.         'This will load the sql procedure GetBooksForBasket
93.         Dim command As New SqlCommand("GetBooksForBasket", connection)
94.         command.CommandType = CommandType.StoredProcedure
95.         command.Parameters.Add("@basketID", SqlDbType.Char, 36)
96.         command.Parameters("@basketID").Value = basketId
97.         Try
98.             'This will open the connecetion to the database which holds the procedu
    res
99.             connection.Open()
100.                 Return command.ExecuteReader(CommandBehavior.CloseConnection)
101.             Catch ex As Exception
102.                 connection.Close()
103.                 Throw ex
104.             End Try
105.         End Function
106.
107.         Public Shared Function GetTotalCost() As Decimal
108.             Dim connection As New SqlConnection(connectionString)
109.             'This will load the sql procedure TotalCost
110.             Dim command As SqlCommand = New SqlCommand("TotalCost", connection)

111.             command.CommandType = CommandType.StoredProcedure
112.             command.Parameters.Add("@basketID", SqlDbType.Char, 36)
113.             command.Parameters("@basketID").Value = basketId
114.             Dim total As Decimal = 0
115.             Try
116.                 'This will open the connecetion to the database which holds the
    procedures
117.                 connection.Open()
118.                 total = command.ExecuteScalar()
119.             Finally
120.                 connection.Close()
121.             End Try
122.             Return total
123.         End Function
124.
125.         Public Shared Function RemoveBookFromBasket() As String
126.             Dim newQueryString As String = String.Empty
127.             Try
128.                 'This will run in order to remove the book the user has selected
     from their basket
129.                 Dim query As System.Collections.Specialized.NameValueCollection

130.                 query = HttpContext.Current.Request.QueryString
131.                 Dim paramname As String
132.                 Dim paramvalue As String
133.                 Dim i As Integer
```

```
                    For i = 0 To query.Count - 1
                        If Not query.AllKeys(i) Is Nothing Then
                            paramname = query.AllKeys(i).ToString()
                            If paramname.ToUpper <> "VIEWBASKET" Then
                                paramvalue = query.Item(i)
                                newQueryString = newQueryString + paramname + "=" + paramvalue + "&"
                            End If
                        End If
                    Next
                Catch ex As Exception
                    Return String.Empty
                End Try
                Return newQueryString
            End Function

        End Class
```

SearchResults.ascx.vb

```
Public Class WebUserControl3
    Inherits System.Web.UI.UserControl

    Private Sub page_load(ByVal sender As System.Object, ByVal ex As System.EventArgs) Handles MyBase.Load
        'This will only load if the user has tried searching for something
        Dim searchString As String = Request.QueryString("Search")
        Dim PageNumber As String = Request.QueryString("PageNumber")
        Dim BooksOnPage As String = Request.QueryString("BooksOnPage")

        Dim HowManyResults As Integer
        'This will get the number of books found relating to what the user has searched for
        HowManyResults = Catalog.SearchCatalog(searchString, PageNumber, BooksOnPage)
        'If there are no results the website will display a message saying that no books were found
        If HowManyResults = 0 Then
            titleLabel.Text = "Your search for <font-style = italic>" + searchString + " had no results"
            pageNumberLabel.Visible = False
            PreviousPage.Visible = False
            NextPage.Visible = False
        Else
            'Otherwise it will display to the user how many results were found
            titleLabel.Text = "Your search for <font-style = italic>" + searchString + " gave back " + HowManyResults.ToString + " results"
            Dim HowManyPages As Integer
            'This will contain how many pages of results are needed
            HowManyPages = Math.Ceiling(HowManyResults / (CType(BooksOnPage, Integer)))

            pageNumberLabel.Text = "Page" + PageNumber.ToString + " of " + HowManyPages.ToString
            'If the user is on page one, 'previous page' option will be hidden and next page will be shown
            If PageNumber = 1 Then
                PreviousPage.Visible = False
                NextPage.Visible = True
            Else
```

```
                PreviousPage.NavigateUrl = "?search=" + searchString + "&PageNumber
" + (CType(PageNumber, Integer) - 1).ToString + "BooksOnPage=" + BooksOnPage.ToStri
ng
            End If
            'If the user is on the last page, the next page option will be hidden i
nstead
            If PageNumber = HowManyPages Then
                NextPage.Visible = False
            Else
                NextPage.NavigateUrl = "?search=" + searchString + "&PageNumber" +
(CType(PageNumber, Integer) + 1).ToString + "&BooksOnPage=" + BooksOnPage.ToString

            End If
        End If
    End Sub

End Class
```

ReadList.vb

```
Imports System.Data.SqlClient

Public Class ReadList
    Private Shared ReadOnly Property rentlistId()
        Get
            Dim context As HttpContext = HttpContext.Current
            'This will check if the user has cookies
            If context.Request.Cookies("RentABook_RentListID") Is Nothing Then
                'If they do not it will create one
                Dim listID As Guid = Guid.NewGuid()
                Dim cookie As New HttpCookie("RentABook_RentListID", listID.ToStrin
g)
                context.Response.Cookies.Add(cookie)
                Return listID.ToString
            Else
                Return context.Request.Cookies("RentABook_RentListID").Value
            End If
        End Get
    End Property

    Private Shared ReadOnly Property connectionString() As String
        Get
            Return ConfigurationManager.AppSettings("connectionString")
        End Get
    End Property

    Public Shared Function AddBooks(ByVal bookId As String)
        Dim connection As New SqlConnection(connectionString)
        'This will call the SQL stored procedure AddToReadList
        Dim command As New SqlCommand("AddToReadList", connection)
        command.CommandType = CommandType.StoredProcedure
        'These are the parameters being passed into the procedure
        command.Parameters.Add("@listID", SqlDbType.Char, 36)
        command.Parameters("@listID").Value = rentlistId
        command.Parameters.Add("@BookID", SqlDbType.Int)
        command.Parameters("@BookID").Value = bookId
        Try
            connection.Open()
            command.ExecuteNonQuery()
        Finally
            connection.Close()
        End Try
        Return 0
    End Function
```

```
44.
45.     Public Shared Function RemoveFromReadList(ByVal bookId As String)
46.         Dim connection As New SqlConnection(connectionString)
47.         'This will call the SQL stored procedure RemoveFromReadList
48.         Dim command As New SqlCommand("RemoveFromReadList", connection)
49.         command.CommandType = CommandType.StoredProcedure
50.         'These are the parameters being passed into the procedure
51.         command.Parameters.Add("@listID", SqlDbType.Char, 36)
52.         command.Parameters("@listID").Value = rentlistId
53.         command.Parameters.Add("@BookID", SqlDbType.Int)
54.         command.Parameters("@BookID").Value = bookId
55.         Try
56.             connection.Open()
57.             command.ExecuteNonQuery()
58.         Finally
59.             connection.Close()
60.         End Try
61.         Return 0
62.     End Function
63.
64.     Public Shared Function GetBooksFromReadList() As SqlDataReader
65.         Dim connection As New SqlConnection(connectionString)
66.         'This will call the SQL stored procedure GetBooksForReadList
67.         Dim command As New SqlCommand("GetBooksForReadList", connection)
68.         command.CommandType = CommandType.StoredProcedure
69.         'This is the only parameter being passed into the procedure
70.         command.Parameters.Add("@listID", SqlDbType.Char, 36)
71.         command.Parameters("@listID").Value = rentlistId
72.         Try
73.             connection.Open()
74.             Return command.ExecuteReader(CommandBehavior.CloseConnection)
75.         Catch ex As Exception
76.             connection.Close()
77.             Throw ex
78.         End Try
79.     End Function
80.
81.     Public Shared Function RemoveBookFromReadList() As String
82.         'This function will remve a book book from the reading list
83.         Dim newQueryString As String = String.Empty
84.         Try
85.             Dim query As System.Collections.Specialized.NameValueCollection
86.             query = HttpContext.Current.Request.QueryString
87.             Dim paramname As String
88.             Dim paramvalue As String
89.             Dim i As Integer
90.             For i = 0 To query.Count - 1
91.                 If Not query.AllKeys(i) Is Nothing Then
92.                     paramname = query.AllKeys(i).ToString()
93.                     If paramname.ToUpper <> "VIEWREADLIST" Then
94.                         paramvalue = query.Item(i)
95.                         newQueryString = newQueryString + paramname + "=" + paramva
    lue + "&"
96.                     End If
97.                 End If
98.             Next
99.         Catch ex As Exception
100.                Return String.Empty
101.            End Try
102.            Return newQueryString
103.        End Function
104.
105.    End Class
```

SignUpPage.ascx.vb

```
1.  Public Class SignUpPage
2.      Inherits System.Web.UI.UserControl
3.
4.      Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Ha
    ndles Me.Load
5.
6.      End Sub
7.
8.      Protected Sub SignUpButton_Click(sender As Object, e As EventArgs) Handles Sign
    UpButton.Click
9.          'This function will only load if the user has clicked on the signup button
    at the bottom of the signup page
10.         Dim name As String = NameTextBox.Text
11.         Dim email As String = EmailTextBox.Text
12.         Dim password As String = PasswordTextBox.Text
13.         Dim rePassword As String = ReEnterPasswordTextBox.Text
14.         'This will compare the string in password to the string in rePassword
15.         If password <> rePassword Then
16.             'If they do not match the program will display the message below on the
     website
17.             SignUpLabel.Text = "Passwords do not match, please re-enter"
18.         Else
19.             'If the passwords to match it will attempt to add the details to the ta
    ble
20.             Try
21.                 Account.AddUser(name, email, password)
22.                 SignUpLabel.Text = "You have signed up!"
23.                 hide()
24.                 'If there is an error returned from trying to add the account, it w
    ould be because the email address is already in use
25.             Catch ex As Exception
26.                 'The website will then tell the user about the email already in use

27.                 SignUpLabel.Text = "This email address is already in use!"
28.             End Try
29.         End If
30.     End Sub
31.
32.     Private Sub hide()
33.         'This function will hide everything on the page
34.         NameLabel.Visible = False
35.         NameTextBox.Visible = False
36.         EmailLabel.Visible = False
37.         EmailTextBox.Visible = False
38.         PasswordLabel.Visible = False
39.         PasswordTextBox.Visible = False
40.         ReEnterPasswordLabel.Visible = False
41.         ReEnterPasswordTextBox.Visible = False
42.         SignUpButton.Visible = False
43.     End Sub
44. End Class
```

RentABook.css

```
1.  .BookTypeUnselected
2.  {
3.      color: blue;
4.      font-family: Comic Sans MS;
5.      text-decoration: none;
6.      font-size: 14px;
7.      font-weight: bold;
```

```
8.      line-height: 25px;
9.  }
10.
11. .BookTypeUnselected:hover
12. {
13.     color: red;
14. }
15.
16. .BookTypeSelected
17. {
18.     color: green;
19.     font-family: Comic Sans MS;
20.     text-decoration: none;
21.     font-size: 14px;
22.     font-weight: bold;
23.     line-height: 25px;
24. }
25.
26. .GenreUnselected
27. {
28.     color: Blue;
29.     font-family: Comic Sans MS;
30.     text-decoration: none;
31.     font-size: 14px;
32.     font-weight: bold;
33.     line-height: 25px;
34. }
35.
36. .GenreUnselected:hover
37. {
38.     color: red;
39. }
40.
41. .GenreSelected
42. {
43.     color: Green;
44.     font-family: Comic Sans MS;
45.     text-decoration: none;
46.     font-size: 14px;
47.     font-weight: bold;
48.     line-height: 25px;
49. }
50.
51. .FirstPageText
52. {
53.     color: Navy;
54.     font-family: Verdana, Arial, Helvetica, sans-serif, sans-serif;
55.     text-decoration: none;
56.     font-size: 12px;
57.     font-weight: bold;
58.     line-height: 9px;
59.     padding-left: 10px
60. }
61.
62. .ListDescription
63. {
64.     color: Black;
65.     font-family: Verdana, Arial, Helvetica, sans-serif, sans-serif;
66.     font-size: 14px;
67.     font-weight: bold;
68. }
69.
70. .SectionTitle
71. {
72.     color: red;
73.     font-family: 'Comic Sans MS';
```

```
    text-decoration: none;
    font-size: 24px;
    font-weight: bold;
    line-height: 15px;
    padding-left: 10px
}

.BookName
{
    color: red;
    font-family: Verdana, Arial, Helvetica, sans-serif, sans-serif;
    font-size: 13px;
    font-weight: bold;

}

.BookDescription
{
    color: black;
    font-family: Verdana, Arial, Helvetica, sans-serif, sans-serif;
    font-size: 11px;
}

.BookPrice
{
    color: black;
            font-family: Verdana, Arial, Helvetica, sans-serif, sans-serif;
            font-size: 11px;
            font-weight: bold;

        }
        .SearchBox
        {
            font-family: Verdana, Helvetica, sans-serif;
            font-size: 11px;
            color:black;
        }
        .SearchResult
        {
            font-family: veranda, Helvetica, sans-serif;
            font-size: 11px;
            color: black;
        }
```

default.aspx.vb

```
Public Class _default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Ha
ndles MyBase.Load
        Dim TypeOfBooksID As String = Request.QueryString("TypeOfBooksID")
        Dim searchString As String = Request.QueryString("Search")
        Dim ViewBasket As String = Request.QueryString("ViewBasket")
        Dim ReadList As String = Request.QueryString("ReadList")
        Dim SignUp As String = Request.QueryString("SignUp")
        Dim AddBooksPage As String = Request.QueryString("AddBooksPage")
        'This if statement checks to see if the user has clicked the Add book butto
n
        If Not AddBooksPage Is Nothing Then
            Dim control As Control
            'If it has it will load the Add books page from the UserControls folder

            control = Page.LoadControl("UserControls/AddBooksPage.ascx")
```

```
16.             pageContentsCell.Controls.Add(control)
17.             'This elseif statement checks to see if the user has clicked the Signup
     button
18.         ElseIf Not SignUp Is Nothing Then
19.             Dim control As Control
20.             'If it has it will load the signup page from the UserControls folder
21.             control = Page.LoadControl("UserControls/SignUpPage.ascx")
22.             pageContentsCell.Controls.Add(control)
23.             'This elseif statement checks to see if the user has clicked the Read l
    ist button
24.         ElseIf Not ReadList Is Nothing Then
25.             Dim control As Control
26.             'If it has it will load the ReadList page from the UserControls folder

27.             control = Page.LoadControl("UserControls/ReadList.ascx")
28.             pageContentsCell.Controls.Add(control)
29.             'This elseif statement checks to see if the user has clicked of the Vie
    w Basket button
30.         ElseIf Not ViewBasket Is Nothing Then
31.             Dim control As Control
32.             'If it has it will load the Basket page from the UserControls folder
33.             control = Page.LoadControl("UserControls/Basket.ascx")
34.             pageContentsCell.Controls.Add(control)
35.             'This elseif statement checks to see if the user has typed in anything
    in the search box
36.         ElseIf Not searchString Is Nothing Then
37.             Dim control As Control
38.             'If it has it will load the SearchResults page from the UserControls fo
    lder
39.             control = Page.LoadControl("UserControls/SearchResults.ascx")
40.             pageContentsCell.Controls.Add(control)
41.             'This elseif statement checks to see if the user has clicked one of the
     type of books, fiction or non-fiction
42.         ElseIf Not TypeOfBooksID Is Nothing Then
43.             Dim control As Control
44.             'If it has it will load the Catalog page from the UserControls folder
45.             control = Page.LoadControl("UserControls/Catalog.ascx")
46.             pageContentsCell.Controls.Add(control)
47.             'Otherwise it will just load first page from the UserControls folder
48.         Else
49.             Dim control As Control
50.             control = Page.LoadControl("UserControls/FirstPage.ascx")
51.             pageContentsCell.Controls.Add(control)
52.         End If
53.     End Sub
54.
55.     Protected Sub ViewBasketButton_Click(sender As Object, e As EventArgs) Handles
    ViewBasketButton.Click
56.         If Request.QueryString("ViewBasket") Is Nothing Then
57.             Response.Redirect("default.aspx?ViewBasket=1&" & Request.QueryString.To
    String())
58.         End If
59.     End Sub
60.
61.     Protected Sub SignUpButton_Click(sender As Object, e As EventArgs) Handles Sign
    UpButton.Click
62.         If Request.QueryString("SignUp") Is Nothing Then
63.             Response.Redirect("default.aspx?SignUp=1&" & Request.QueryString.ToStri
    ng())
64.         End If
65.     End Sub
66.
67.     Protected Sub AddBookButton_Click(sender As Object, e As EventArgs) Handles Add
    BookButton.Click
68.         If Request.QueryString("AddBooksPage") Is Nothing Then
```

```
69.             Response.Redirect("default.aspx?AddBooksPage=1&" & Request.QueryString.
    ToString())
70.         End If
71.     End Sub
72.
73.     Protected Sub ReadList_Click(sender As Object, e As EventArgs) Handles ReadList
    .Click
74.         If Request.QueryString("ReadList") Is Nothing Then
75.             Response.Redirect("default.aspx?ReadList=1&" & Request.QueryString.ToSt
    ring())
76.         End If
77.     End Sub
78. End Class
```

References

http://python-textbok.readthedocs.io/en/1.0/Sorting_and_Searching_Algorithms.html

http://vb.net-informations.com/communications/vb.net_smtp_mail.htm

https://github.com/

ASP.NET for Dummies (book)

http://www.visual-basic-tutorials.com/

https://stackoverflow.com/

https://www.tutorialspoint.com

VB.net for beginners (book)