**COB295 Team Project Report**

By Ryan Lewis, Coral Punchard, Robert Tamcken and Simon Yip

Task

The aim of this project was to successfully work in groups of 4 to create a web application called iBay, which is to function as an auction website. Through this, the teams were expected to create a login page, search page and upload page where each page has specific requirements which shall be discussed in the methodology. The main focus on the functionality of the website should have been focused around the search page where the user should be able to search for any item and a list of results should be displayed, and the other main focus on the upload page where the user is able to describe a new item for sale and upload images with this, provided they are a registered user of the website.

Moreover, each group has access to a database in which the information can be stored for use in the website. This database consists of three tables: items, members and images.

All through this, uses of HTML, CSS, Javascript, jQuery, PHP, JSON, AJAX, MySQL languages should be shown to be used in good confidence.

Definitions

HTML - Hyper Text Markup Language. This is a series of tags which are used to describe a web document. The browser then reads the document and uses the tags in order to work out how to display the information provided.

CSS - Cascading Style Sheets. This works in tandem with HTML as this describes the presentation of the HTML document. e.g. the colour of the text, the background colour etc. This effectively is used to style the webpage, which is to say, how each page will look when on display through a browser.

JavaScript - Javascript is the programming language for HTML. This means that through this, you can create functions which can do specific things, e.g. when the user clicks a button (created through HTML), the current time can be displayed. There are many applications that javascript can do.

jQuery - This is a Javascript library in which simplifies the Javascript programming through handy functions already available through this library. With the use of this, manipulation, event handling and AJAX are much simpler to use.

PHP - Hypertext Pre-processor. This is a server side scripting language which can be embedded into HTML script using php tags. This acts primarily as filter taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data.

JSON - Javascript Object Notation. This is a lightweight data- interchangeable format which works mostly with a collection of name/value pairs (object, record, associative array etc) and also ordered list of values (arrays, lists etc).

AJAX - Asynchronous JavaScript and XML (XML is similar to JSON). This allows web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behaviour of the existing page. Ajax allows for web pages to change content dynamically without the need to reload the entire page.

MySQL - This is one of the most popular open source relational SQL database management systems. Through this, a database can be edited but also information read and used by the web application in use. To use this, it must be through PHP.

With various languages being used within this project, it is useful to know which ones are client-side and server-side as this can aid in the efficiency of the coding and the working of the web pages as a whole.

Client side languages: HTML, CSS, Javascript, jQuery, JSON.  
Server side languages: PHP, AJAX, MySQL.

Client-side vs. Server-side Programming

Introduction

In terms of web development there exist two main types of programming languages; server side programming and client side programming. Websites run scripts either on the client side, sometimes known as the ‘front-end’, or on the server-side, sometimes referred to as the ‘back-end’.

The difference between the two that can be seen from a user perspective is the fact that client-side is what you can see on screen. This means that all the texts, headers, links, buttons are created through client-side programming. Moreover, when the user then clicks on one of these e.g. a submit button, functions will run in the background to submit the data inputted. The functions that run in the background (which the user cannot see) is what is known as server-side programming.

More specifically the client refers to the party that requests pages from the server and displays them to the user. Normally this will be a web browser such as Google Chrome or Internet Explorer, i.e. the client is a web browser.  Furthermore, the ‘server’ is a web application server at a remote location that will process web requests and send pages to the web browser (i.e. the client).

Client-Side programming

The most widely used client side programming language is JavaScript but there are other languages such as Actionscript  which is commonly used in the creation of mobile, desktop and web applications, interactive animations and video games which is run through adobe flash. The client side languages are used mainly for dynamic and interactive page updates and contents. The reason these languages are called client side is because they run scripts on the user’s computer after the web page has been loaded from the server.

Advantages of client-side programming

Client side has a number of advantageous uses, for instance, it can make webpages interactive and allow for dynamic changes to a website. For example, a website that aids in teaching children mathematics, proposes a question for the user to work out. The web developer doesn’t want the user to see the answer until they have attempted the question themselves. To do this, they can add dynamic content such that when the user presses a button, the website displays the model answer underneath the question. This is all done without need of a page refresh and hence leads to a more pleasant and smooth experience for the user. This is beneficial because it creates a faster response time for the web page but also means that the server is not contacted as the action is processed on the user’s browser.

Moreover, it is often used for validation of an input. For example, in our website, the bid box on the Item Description page uses a JavaScript function validateBid() that does exactly that, it validates the user’s bid. It checks to see if the inputted bid is of a high enough amount (i.e. is the value larger than the previous?) and to ensure that the user is logged in, in order to place the bid on an item. The use of validation on the client side is very important to reduce the necessity for redundant server calls, followed by server returns of an error message, especially when speed is a critical factor, as the user will want a quick response and not be waiting for the page to reload.

There are many functionalities that client-side can do such as using CSS, applying styling onto a webpage, for example, the background colour or text font. Moreover they can aid in making calculations without the need for page refreshment which as stated before, helps in increase the response time of the web page. Also, through scripting languages such as ActionScript, animations can be created and implemented onto the webpage.

Disadvantages of Client Side Programming

The main disadvantage of using client side languages is the fact that the scripting languages which one has to use in order to implement client side, are often more complicated and require more time and experience to programming the scripts correctly compared to those classed as server-side languages. Looking at JavaScript compared with MySQL, MySQL is very simple and easy to learn having only a small selection of functions such as SELECT, UPDATE etc. Although JavaScript is also easy to learn, it takes more time to fully understand the syntax of the language and is much more in-depth than the contrasted MySQL.

Moreover, the scripting languages that one chooses to use must be supported by the web browser in which the webpage will work off. If the web developer uses an unsupported scripting language, then the functions of interaction will fail and the user will be left with error messages on the webpage. When developing a web application, it is very important to consider which browser it will use as this will have an impact on which scripting languages one can use within the main programme.

Furthermore, when viewing the source code for a page, the user can edit the code and hence change the functionality of the website as a whole. This gives way for a security issue in which cannot be maintained. Only through firewalls can the code be protected from changes and hacks.

Also, client side can only store data for as long as the page takes to reload. The only exception to this is when a browser takes a note of useful details such as your login details.

Server Side Programming

Server side programming runs its scripts before the HTML is loaded. This is where there is interaction with permanent storage such as a database or files.  The scripts are run on the server which hosts the website and sends through the HTML code to the client, normally a browser, to display and hence when you view a page source code you do not see any server side script, as it has already been used to create page content. One can check this by going to any webpage, right clicking in a blank space on the website, and from the menu that appears, clicking ‘view page source’.

There are many uses of server side programming, much like client side, however server side specialises in such things as processing user inputs and displaying pages as well as customisation of that webpage in which the user is operating.

This follows to one of the main uses for server side programming, which is to interact with a database (a remote, permanent storage facility).

Advantages of Server Side Programming

In particular, it is very beneficial to fetch results from a database after running a query to display particular information. For example, if the user would like to change their profile information in a web application registered account e.g. Facebook, at first, it would display the current information held within the database. Then, once the information has been updated by the user, the user will then submit this page where the request will go to the server and the query of UPDATE will have taken place. Hence, the webpage will now display the new, updated information to the user once the page as reload. The fact that the page has to reload in order for this to be successful is one of the key qualities that shows that it was a server-side action that took place. Normally a reload function is put in place within the server side scripting in order to make it a smooth experience for the user. This makes server side a brilliant way to store values for a long period of time such as a user’s email address and contact numbers.

However, many server side scripts need to have access to a database in order to store the dynamic data. If a developer chooses one that cannot do this, then a lot of problems will occur in being able to store the data and also being able to manipulate the data.

We also use PHP (a well-known server side language) to pass through variables from one page to another and again use PHP on the destination page to request the information passed through. For example, we can pass through form data using GET or POST methods.

If there is a very complicated and client-side focussed web page code, then the user may find that the load time increases. This is due to the amount of processing having to be done by the user’s own browser. In order to combat this, using server side code for the same functionalities as the client side can increase the speed of the page loading and therefore reload the page in a much quicker response time. The reason for this, is that the server side has no need for the plugins that one has to install such as Flash and Java.

It is also important to use server side programming to encode data from server side programming languages into client side programming languages. For example, if information fetched from the database is stored in a PHP array, then we would like to encode it into a client side, for example JavaScript, array, such that we can use the information within the HTML code on the client side and not on the server side before the page is sent to the client. This is done for efficiency. Moreover, if this is not done, then that the information/data will not be shown on the webpage as what is shown is only in client-side languages. Therefore it is important to use JSON.encode() to enable the information to be read by the browser in an acceptable language (here it would be JavaScript).

Disadvantages of server side programming

One of the main disadvantages to server side programming is the threat of over demand of a website. This is especially a concern for websites with large applications and/or with large need to extract data from a database to display the site. If there is large traffic to the site, the demand on the server to load and send these large pages to many clients could be too much for a regular server, especially if the website owner has only used a small server due to not expecting large traffic. This could result in very long load times and hence bad user experience for those trying to access the website. The website owner may need to look to dedicated servers or cloud hosting to try to get around this problem, but both could be expensive solutions. However, the website creator can use their own judgement to know if these will be problems for their site or not.

Secondly, server side programming that is used for dynamic content will require a page to be refreshed to see the change in content. For example when a user logs in they will see a page refresh. However it is worth noting that with Ajax, this is becoming less of a problem and can be used to display updates to the page without the need for refreshing.

There is also a security issue with using server side programming. The use of server side programming could give anyone with malicious intent, hackers or criminals, the opportunity to access the servers by exploiting poor coding or coding oversights. This may be particularly worrisome for large websites that store lots of sensitive user information such as credit card details and addresses. There could also be a risk of backdoors occurring from either poor coding or deliberate programmer intention. It is dangerous as it could allow malicious code/content to be injected into the server and could be used to affect every computer of every user of the website or be used to get database information.

Furthermore, there is a possible threat to security due to poor password protection implementation in server side scripting. A hacker may try many different words to guess a password for a user, sometimes known as ‘dictionary attacks’ and server side scripting has no way to detect this normally.

Conclusion

In conclusion, throughout this we have discussed the benefits and drawbacks of both client side and server side programming. It is very common that websites need to make use of both client and server side languages. Some things can solely be done using the specific languages, but others can be done with either type of language. However, it is important to use the type of scripting which will result in the most efficiency and for the best client experience.

Client side should be used for anything that may require user interaction and validation that will be used to ensure no unnecessary calls to the server, while server side should be used wherever it is necessary for dynamic loading of data to display to the website. Therefore, when deciding on which languages to use for your own web development project, it is important to think about the functionality of the bit you are doing and think what applies to it best. For instance, if a web page needs to display a user’s favourite colour, this information would have to come from a database. Therefore, you would have to use server side in order to get the data from the data and display it on screen. However, if you’d like to create a button that would display a message, this could be solely accomplished through client side as there is no need to access a database for the information needed.

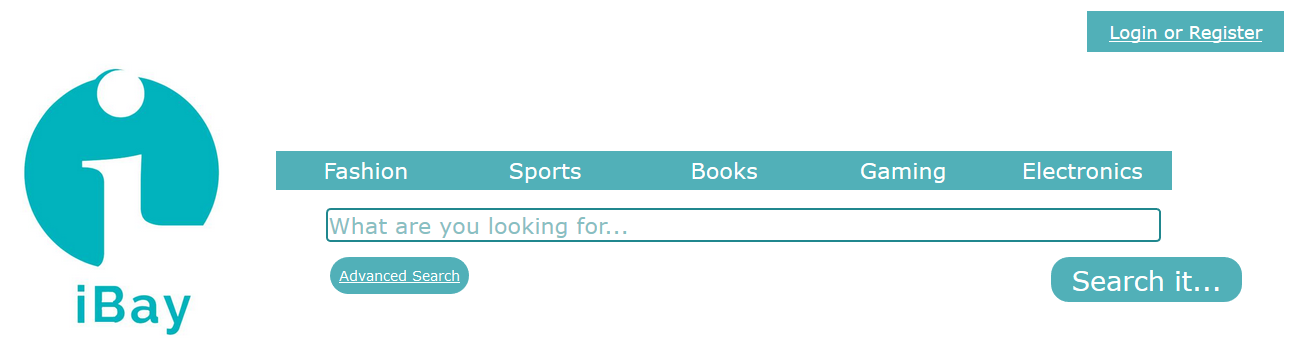
**Methodology for the webpages**

Contributions:

* Homepage: Simon Yip and Ryan Lewis
* Search Results: Simon Yip
* Advanced Search: Coral Punchard
* Items Description: Ryan Lewis
* Login: Simon Yip
* Upload: Robert Tamcken
* User Profile: Coral Punchard and Ryan Lewis
* Database: Coral Punchard
* Database Connection: Simon Yip

**Homepage**

The screenshot below shows the Homepage for iBay.

****

To left you can see the iBay logo, which is hyperlinked such that when pressed (on any page, not just the homepage) will redirect the user to this Homepage.

In the centre of the page, at the top, you can see a navigation bar containing the five main categories of items that are listed on iBay, namely Fashion, Sports, Books, Gaming and Electronics.

Below this is a search bar. This is designed for basic searches. It allows users to press the box and then type in what they wish to search for. They can then submit this search by pressing the “Search it…” button to the bottom-right of the text box. This will then redirect the user to the Search Results page.

Furthermore, there is an “Advanced Search” button. This allows the user to be more specific in their search, for example allowing specification of price range of the items returned in the search results or searching for only items within a certain category. Hence, when the user presses this button, they will be re-directed to the Advanced Search page.

In the top-right corner of the page is the “Login or Register” button. This obviously allows the user to log in to their accounts if they have one, or register with the site if they do not. Hence, when pressed, it re-directs the user to the Login page.

If the user is already logged in the buttons below will be visible instead of the “Login or Register” button:

https://lh6.googleusercontent.com/rGMDplndOSXaM8ehvPaXF1IeJ1wYSEDPrBO6wTGkzlS_vVlII0hMgl-uHTcTHEx2DGCPRHPnNDaWiDphoeyzXyNBZFzwD7a7RXqEg2vPuTnuHkGwjKlsUhd0GYAWBZL4PEOKktHAM2j722qAWg

The “Your Account” button, once pressed, will link the user to the User Profile page where they will be able to see and edit their account details, as well as see and edit the items that they are currently selling and the items which they are currently the highest bidder on.

The “Logout” button logs the user out of their account. The user is then redirected to the Homepage as shown above. These buttons are present on all pages and work in the same manner on all pages.

The page uses some Server Side programming for two buttons. First, it checks $\_SESSION variables to see if they are set and hence the user is logged in to their account. If they are, then as stated before the HTML will output “Your Account” and “Logout” buttons, but if not the HTML output will be for a “Login or Register” button.

The page also uses PHP to check if the ‘Logout’ has been pressed and if so, the PHP code also destroys session variables and if “Your account” is pressed, it will run a php code that will redirect the user to their user profile page.

The only other code featured in the page is HTML and CSS.

For example, the buttons all use CSS such that the colour of the button changes when the user’s mouse is hovering over it. The effect can be seen below:https://lh5.googleusercontent.com/CwACq7A_qKA1Rl9DC5WHJd2nkycimz71prDr1LkSPE67iaL5psoDqsiehSRsmN7jdCOm4txq-0mOpJJuM1fSE2zkHhmon_rzdidWIqJbMVigWJi2qHYHMzWyYBzdR-KAPQqizacF  


The top screenshot shows the navigation bar with no mouse over effect, while the bottom screenshot shows the navigation bar when the user’s mouse was hovering over the “Fashion” button.

**Search Results**

This page takes variables from either Homepage, AdvanceSearch, whichever page the user was previously on, to construct the sql statement which will query the database from these variables and output the query results onto the page. The server side (PHP code) are mainly for getting the variables from the previous page and the client side (JavaScript and HTML code) will sort the data returned from the database following the query request and display them in an orderly fashion.

How does the page work (server side)?

The first 10 lines of the php are for when the user log out and when the user want to go to their user profile page. (See homepage for details)

First, using the this ‘if’ statement if( isset($\_REQUEST(‘*nameofvariable’))*, the php will check which form is submitted to this file. The possible search types are

* The simple text search (the submit button is named: search\_input)
* The advanced search (named: advsearch)
* The category search (named: search\_default).

$\_REQUEST() function will request the variable that is submitted through the form and isset() function checks whether it is NULL or not. If the name of submit variable we are requested exists, then we know which type of search it is.

Category search

The search\_default form is the five category links that are located in navigation bar of every page in this website except the login page. The categories are: fashion, sport, book, gaming and electronics, as you can see from the screenshot below. https://lh5.googleusercontent.com/Ml3tgEcvbPaOLBrN-lpPF3Vwso9q4mw_1di7a1Rw4XsY7zK9GlZv2UzwPNLsBbwpW1lmDmMWQU18V-2Djo5iwotcJ3ZFgOPVwZonHABnks7ZLkf4mcLanNpSbB8-mRz7VtGTX1P8

These values are named as search\_default, so when you select one of the category, the selected value will be passed down to SearchResult page and we could obtain this value by using the $\_REQUEST(‘search\_default*’*). Since the value are attached to the submit variable, we only need request the submit variable which will contain the category the user had selected and construct an sql statement for it. The sql statement for this search is as follows:

"SELECT \* From iBayItems LEFT JOIN iBayImages ON iBayItems.itemId = iBayImages.itemId WHERE Mainimage = 1 AND category = '".$search."' AND finish > now() ORDER BY iBayItems.itemId DESC ;";

$search is the value from $\_REQUEST(‘search\_default*’*). This sql statement will select all the fields and items from two tables, iBayItems and iBayImages with the following conditions:

* Mainimage field is equal to 1, which indicates it is the main image for this item
* Category equal to the value the user has selected (i.e. fashion, books etc)
* Finish > now() will select all the items that are live.

The results are ordered by newest upload in the database (see lines 15-20 in the file). The sql statement will be passed down to the javascript section of the page which will be discuss later in the report.

Search Bar

Next, the simple text search is a search bar where the user can type in the keywords or title of the item they are looking for. The search bar is found below the navigation bar for the homepage and above the navigation bar for every page except the login page.

https://lh6.googleusercontent.com/dLNLZD6lMsbS9URxkP_LxBM76V-s35AWZ6GGd4dfAn4LGNCU36zHPUULq4-v8vBCG90oNjdFLa21cInszjRkGi0GVjmz6iSWLcTnvPxYi-33khp2Nus37zn-Um8XedeyzEPdKuzp

When the user click on the search button, the string is stored in the variable search\_input, so if $\_REQUEST['search\_input'] is not NULL, then it implies the user is using the search bar to search (line 23).

Once we know the user is searching from the search bar, first we store the string from the search bar (refer as search string) into a variable. Then, we check if the string has any redundant spaces at the end and we remove them from the string. To do this, we compare the last character of the string. If the last character is a space, we will shorten the string and continue doing so until we see an alphabetical character at the end of the string. (See lines 26-38 in the file for specific details)

Next, we proceed into breaking the string into words and sentences. This is to maximise the quality of the sql statement, to make sure we are searching words or sentences are that related to the item. The string will be broken down into three array, individual words, sentences formed from taking words out from the left and right of the string. This can be done from taking the location of spaces and splitting the string words by words from each side. (See lines 43-63)

Finally, we merge these arrays together to form the final array which has all the keywords, parts of the string and proceed in adding these array elements into our sql statement. Similar to before, the sql statement begins with selecting all fields from iBayItems and IbayImages where they have same condition except the category condition is not the same as before.

Now, we have two additional condition to limit our sql select statement:

* If the **item titles** are similar to any of the array elements, they would be selected.
* If the **categories** are equal to any of the array elements, they would be selected too.

To write this into our sql statement, we will first concatenating the first element manually to the sql statement.     $sql = $sql."AND ( title LIKE '%".$result[0]."%'";

This is to start the new condition with the AND and the bracket. Then, we loop through the array and adding the new condition for title and category as OR,   
i.e.  $sql = $sql."OR title LIKE '%".$result[$k]."%' ";  and continue concatenating new conditions until the end. (See line 71-84). (Maybe more later)

An example of an sql statement for this search would be:  
SELECT \* From iBayItems LEFT JOIN iBayImages ON iBayItems.itemId = iBayImages.itemId WHERE Mainimage = 1 AND ( title LIKE '%PS4 XBOX%' OR title LIKE '%PS4%' OR title LIKE '%XBOX%' ) OR ( category = 'PS4 XBOX' OR category = 'PS4' OR category = 'XBOX' ) AND finish > now() ORDER BY iBayItems.itemId DESC ;

Advanced Search

Lastly, for the advance search, it works very similarly to the search bar but with a lot more conditions applied to the sql statement. First, we extract all the variables from the form including how the result is sorted before we start adding the sql conditions. (lines 86-97, see advance search page for the screenshot of the form). Then, using the initial sql statement above (with category added if it is selected), we proceed in concatenating sql conditions if the user has inputted or selected the fields in the advanced search.

For the search bar in the advanced search, the user is able to manipulate how they want their keywords to be searched in the database. (More information about this in the advanced search section). If the user chose a normal search, it will run the same piece of code as the search bar but category conditions are not included for the keywords. For exact search, it will find an item that matches exactly with the keyword. Finally for starts with and ends with search, it will find an item that matches either the first part or the last part of its title with the keyword (lines 162-178). After adding the conditions for the keywords, we close the sql statement and this brings us to the end of server side processing for now.

How does the page work (client side)?

Our client side is consist of manipulating the data obtained from the database and outputting the html code to display the items on the page.

First, we will get the sql Select query and the sort variable from the php code. Now using $(document).ready(function(){…}); jquery function, code within the function(){…} will run after the document is loaded and php code is finished. To get the data from the database using the sql statement, we use the $.get jquery function to get the query results (More details in the SearchQuery section). The query results will be returned as a JSON Object Array.

If we don’t get any result, we will write the html code displaying no matching results as a string in a javascript variable and use .append() jquery function to add this string into the itemtable which is coded in the html (lines 192-209).

Otherwise, we will copy the results into two arrays, named Original and JSONitems, and create the html code for the rows to display the query results using a function that I defined in the javascript, createtable(). Createtable function takes in the JSON array, iterates through the array, creating the html code to hold the item data for each iteration, appends the html code with .append() jquery function. The jquery function $.each will iterate through JSON array, placing each item data into their respective position in the html table cell (lines 211-224).

Other javascript functions we have in this page are:

* Itemlink

This will copy the itemId of the selected item from the itemtable to the hiddenform called itemresult. Then it will automatically submit this value to the item description page. The user will be redirect to this page where they can see the detail of the selected item.

* TimeRemains

This function is the work out the time remaining for the item. It takes the finishing time of the item and subtracts it with the current time. This function can return the number of days, hours, minutes, seconds and milliseconds (named as total).

* sortTable

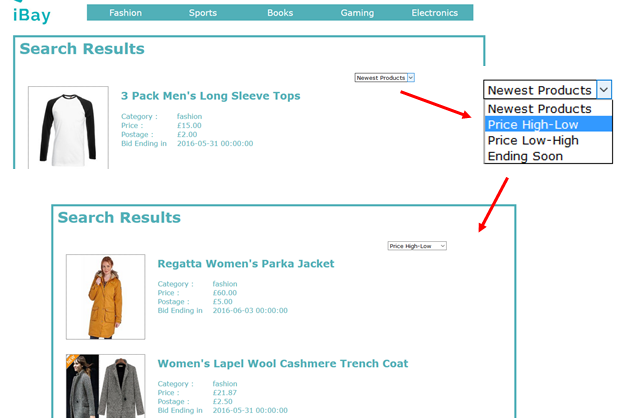
There are four sorts available for the data, NewestProducts (by default), Price from high to low, Price from low to high and EndingSoon. For each sort except NewestProducts, we iterate the JSONitems array, use bubble sort to sort the order of the items, remove the current existing items displayed and add and display the sorted items to the page. With EndingSoon sort, we will using the TimeRemains function to sort the item with the least time remaining in the sort.For the NewestProduct, we simply remove the current items displayed and append the Original array which has the same initial ordering.

How to use the page

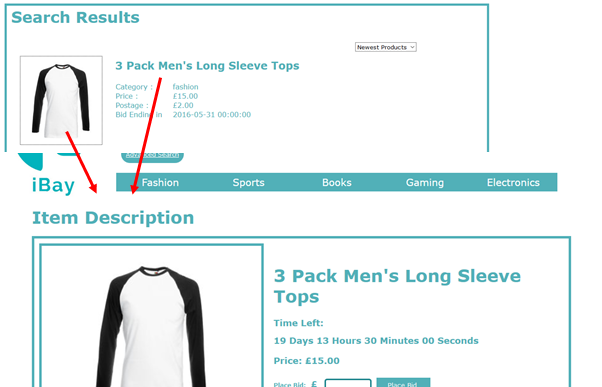
When the items are displayed in this page, the action the user can take are:

* Navigate to:
  + Homepage by clicking on the logo
  + UserProfile page if the user is logged in into the webpage
  + Login page if the user is not logged in
  + Advanced Search page by clicking on the advanced search button
  + Item Description page, which can be accessed by clicking on the title or image of the item displayed in the table.
* User can:
  + Log out from the webpage, by clicking the log out button if the user is logged in
  + Use the search bar to generate a new search
  + Click on the navigation bar to do a category search
  + Sort the items using the dropdown box.

To use the sort functionality of this page, the user just needs to simply select from the drop down box for the sort they wants. The screenshot below the items before the sort, select a different value from the dropdown box and the result that appeared in the form.



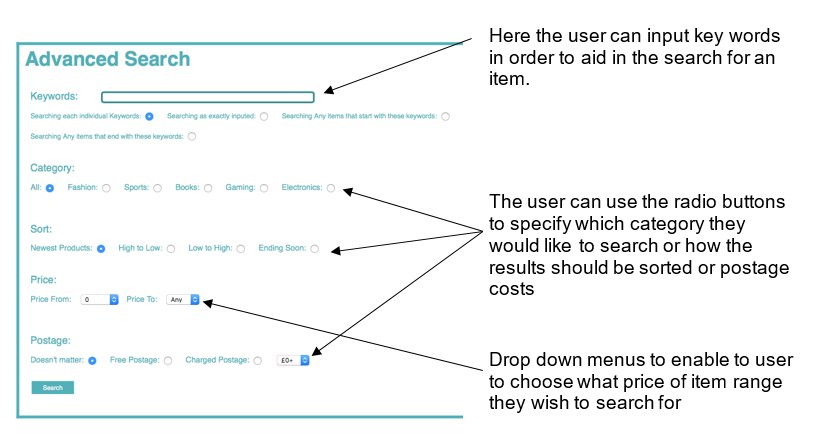
As you can see, by changing the drop down box to “Price from high to low”, we can see that the order of the displayed items have changed to displaying the highest price at top and describing to the low price.

To access the item description page of a specific item, the user simply need to click on either the title or the image of the item displayed in the page. The screenshots below shows the user clicking an item from the page and redirected to the item description page of that specific item.  
  
  
  
All the functionality I have mention works in this page.

For future improvement, I would like to limit the number of item displayed in this page and have buttons to navigate between the item results. The reason this functionality was not implemented is because our database currently does not have too many items to display. This functionality would be useful when the database has an enormous amount of items to process. Also, it would be ideal if each item we displayed in the page to have the time left of the item counting down from the current time.

**Advanced Search page**

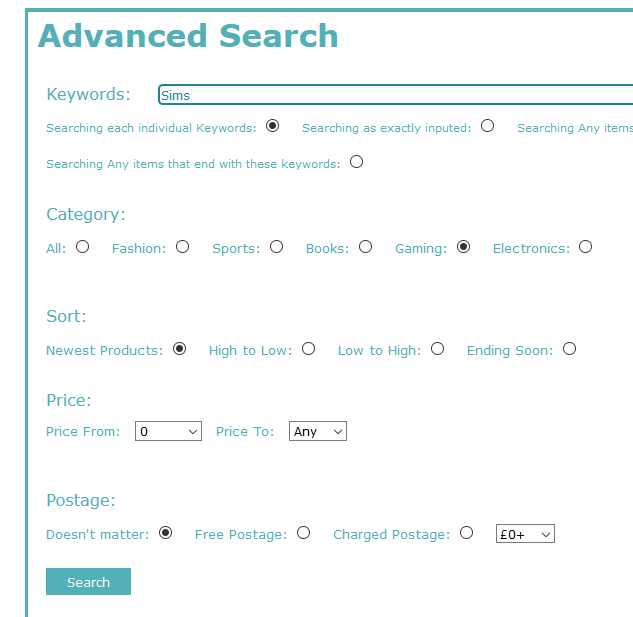
The advanced search page is used by the user to define a more accurate search in order to find a specific item.



How to use the page

This page is one of the simplest throughout the website to use. The user only needs to put any key words, and extra refined search.

For example searching ‘The Sims 4’ under ‘Fashion’ will bring up the message of ‘No results found’ as this is obviously invalid searching criteria. However, if the use were to enter ‘Sims’ into the keywords box and select ‘Gaming’ from category, the result would show only ‘The Sims 4’ PC game.





https://lh3.googleusercontent.com/U8Z9e8ZPSoLzWqhamtIKJghnz7CHhrJ_YXEX4pQwd9CQL3RYB7kaTZiE_F9yEp561f6kcXb-kil5vh_bvo6eJgKP2TynCDJfv9QOSwjWYtM3Yvzq-oezQzxrWJ5c6O4nuYXPOwF2

How does the page work? (Server side)

Within this page, there is very little server side programming done due to the nature of the actions of the page. There is no need for any information from the database to be gathered.

At the beginning of the code, there is the PHP code in order to get the session variables in order to see if any user is logged in. There are also two if(isset.. loops if a logged in user clicks on either ‘Your Account’ or ‘Log Out’.

How does the page work? (Client side programming)

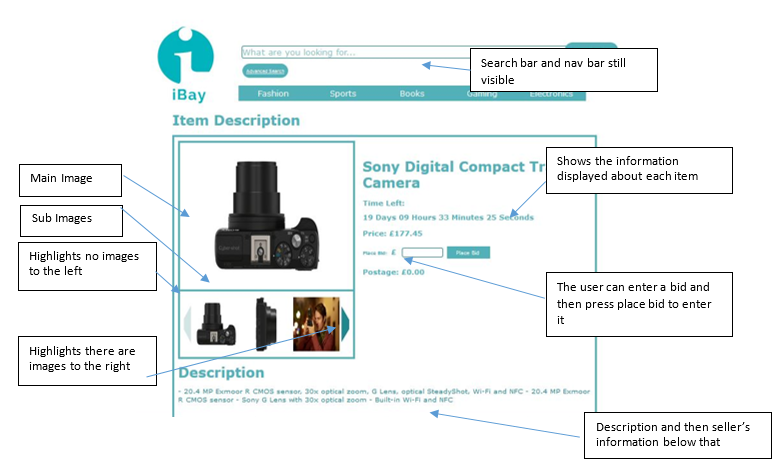
The framework for this page is all written in basic HTML as no complicated function needs to take place.

In order to complete this, one first creates a table, then a form and then another table.

Then the process is very repetitive of creating input tags with various types such as radio, text and submit. For the price drop down menu, one uses a select tag and then option tag.

If the user does not change any of the settings and clicks submit, then it will return all possible items the user can bid on and the sort automatically does newest product first.

**Items Description**

****

How it was created (Server Side)

This particular page requires use of server side, PHP, and client side programming. The server side is used at the top of the page to interact with the database. It is used to store strings in variables that hold SQL queries that are used to get information from all three tables, iBayItems, iBayMembers and iBayImages when we use PHP to send the query to the database. We then, if no errors were returned in which case we would kill the request and output the error message, store the results, if there were any, into PHP arrays and then json\_encode these such that we can use their values within the client side programming.

Firstly, the use of PHP is need to reference any variables that were passed through from previous pages using the $\_REQUEST method. For example, my page needs to reference the ‘itemId’ variable of the item link that the user pressed in either the Search Results page or the User Profile page such that the ItemDescription page can call the database for and display the relevant information regarding that item. Furthermore, we use PHP to reference $\_SESSION variable to check if the user is logged in.

This is necessary to check if the user is allowed to place a bid, and then for when the iBayItems table ‘highestBidder’ field is updated.  Furthermore, we use PHP to see if the ‘place\_bid’ variable was sent through the page using the $\_REQUEST method. If so, we know that the bid has been validated and that a user is logged in due to the JavaScript validation function that will be discussed in the client side programming further on. Hence, the code sets up a string variable to hold an UPDATE SQL query. Server side scripting then sends this to the database and, as long as no errors were returned, updates the price and highest bidder fields of the item being viewed.

How it was created (Client Side)

The client side programming, JavaScript, starts by declaring numerous variables which will be used throughout the page. In particular, we declare JavaScript arrays to store the values that were encoded from the PHP arrays which contain all the information required from the tables in the database, namely the images of the item, the item information and the seller information arrays. This is very useful, as throughout the HTML code in the page, we need to refer to these information values numerous times, and hence it is much more sensible to call these values from a JavaScript array than to keep inserting PHP code such that we can reference a PHP array for the reasons discussed earlier.

Hence, this page uses more client side programming than server side. We have ensured that all parts of the code that can possibly be client side programming actually does use it, again for the reasons previously mentioned.

The first function calculates both the current time and end time in milliseconds. It then calculates the difference in terms of how many months, days, hours, minutes and seconds and is then returned such that we can dynamically display a clock which counts downwards.

If the item has not yet finished its auction then we display the time difference in the form ‘months days hours minutes seconds’. Furthermore, the JavaScript function allows the page to dynamically update the Time Left, and hence it appears as a counter and counts down the time left while the user is still on the page. This is very useful for when the current time is very close to the items finish time as it ensures that the user doesn’t have to keep refreshing the page to see how long is left.

However, an *if* statement is used within the time function which checks to see if the difference in milliseconds between the current time and end time is equal to or less than zero, hence indicating that the auction has finished. Then, the JavaScript function will remove the time, and replace with the text “ITEM ENDED”. Furthermore the bid box and bid button are removed from the page too.

This is necessary to ensure that if a user is looking at the item while it ends they would still be able to enter a bid and so change the database price and highestbidder which would be erroneous. Hence, by removing these elements from the page dynamically with JQuery, we remove any such circumstances.

The bid box also has use of JavaScript functions. These particular functions are used for validation. The first validation ensures that any bid entered is larger than or equal to the current price plus 5%. If not, there is an alerted message to the user to tell them of a bad bid. The second validation is used to ensure that the user is logged in to an iBay account.

This is important as otherwise the database would be updated with a new price if a valid bid value was entered, but we could not update the highestBidder field. Hence, the function alerts the user to the necessity to log in to place a bid. This was done on the client side as it is very simple validation and is likely to be used often if the website was to be very busy, and hence would mean lots of calls to and from the server if we were to have done the validation on the server side.

We then create JavaScript variables that will store strings of HTML code. Then we append those strings to the areas of HTML code where they should appear. This was to improve the readability of code as opposed to having lots of HTML code.

We have a function which creates a table to store the sub images. It checks to see how many images are going to be displayed. If it is more than four, it only displays four and if it is less it displays blanks in the remaining table data items.

We also have JavaScript functions which change the main image and change the sub images which are visible. The function that changes the main image uses JQuery to get the area with id=”mainimgdata” and sets its source equal to the source of the image that was passed to the function (i.e. the image that was pressed). The sub-image changing function checks to see if the direction of change is right or left and then checks to see whether there are actually more images that can be displayed in that direction. If so, it creates a new table of images and displays it to the page by the function that creates the sub images.

At the bottom of the code, we have pure HTML code. This is used to set up the layout of the page and create all the divs and ids which the JavaScript functions that run on page load will append to.

How to use the page

This page can be accessed from the SearchResults page where the user clicks on the item or from the UserProfile page where the user clicks on the item.

On the top left-most side of the page there is a section devoted to housing the images. This is split into two sections. There is the main image, which is the largest of the images and is useful for viewing images for clarity.

Below this there is a section that represents a catalogue of all images, restricted to showing only three at a time however, that represent the item that the user has selected to view from the Search Results page.

In terms of functionality, the user can press any of the images within the images in the catalogue shown and then the main image above will change to show the selected image. This is useful for the user to see an image in larger format and hence slightly more clarity.

Furthermore, there are two arrows on either side of the image catalogue. These arrows allow the user to move the catalogue of images along to give the user the opportunity to view all images that the seller has uploaded to represent that item. The left-most arrow moves the image set by one to the left, and the right-most arrow moves the image set by one to the right.

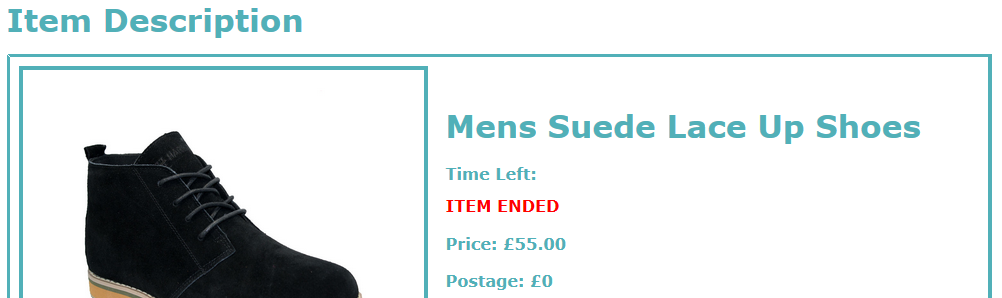


The arrows may either be a dark green colour, which signifies that there are more images to be displayed in that direction, or faded green arrow, which signifies to the user that there is not. Hence, the image set will only move if a dark green arrow is pressed.

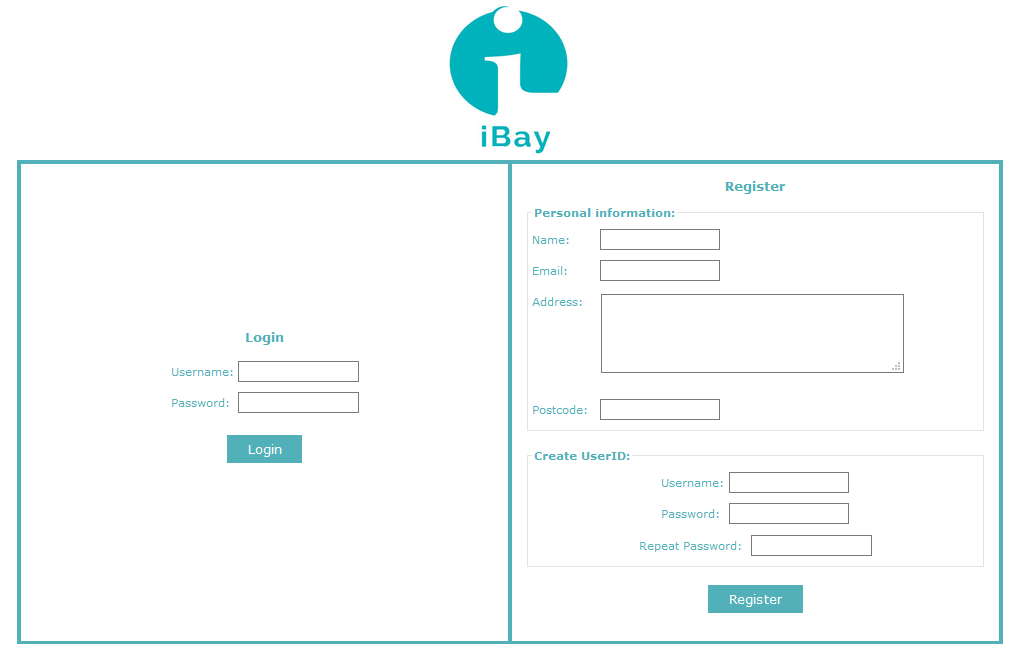
Next to the main image, the information about the item is displayed as well as an area for the user to enter a bid. The bid will work if the user are logged in and pass a suitable bid value (discussed further on). The information displayed in order moving downwards is: Item Name, Time Left until the item auction has ended, Price (current), Place Bid box, Place Bid button and finally Postage cost.

The functionality of the bid box allows users to enter a numerical bid into a textbox, which is part of a form within the HTML code, and then press the ‘Place Bid’ button to the right-hand side of this text box to submit their bid. This, once validated for correctness, is then updated into the database, updating the price and highestBidder of the item.  
The screenshot above shows the message that pops up to the user when they try and enter a bid but are not logged in to an iBay account.

Below the images and the main information section there are two sections, the top-most dedicated to displaying the item description and the bottom-most displaying the seller’s information including: name, address, postcode and seller rating.

If the user is accessing this page through the user profile and the item is finished, then the “Time Left until the item auction has ended” will display “Item ended” in red font instead of the countdown timer, as you see below.  


**Login Page**



The page consists primarily of a table containing two forms, one for registration on the site and one for logging in to the website. Completing either of these forms will redirect the user to the homepage.

How the page it works

The page consists of a table containing 2 forms, side-by-side. All of the fields in a form must be filled in, else the form will not be submitted and an error message will be displayed on the page. Each field's validation is handled using jQuery and has an error message associated with it.

Upon pressing 'Register', the values within the form fields are validated by comparing the username against the database (using PHP to assign variables and MySQL to execute queries). If no match is found, the fields are checked to ensure they are not empty. If they are not, the user ID is added to the database and the user is logged in. If a match is found, the user is prompted to change their desired username and re-submit the form. The same checks are repeated until a valid unique ID is created.

Upon pressing 'Login', the form data is compared against the database. If a valid match is found for both username and password, the user is logged in.

In both successful cases, the user is then redirected to the homepage.

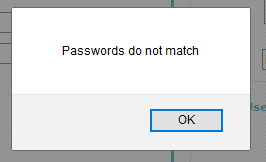
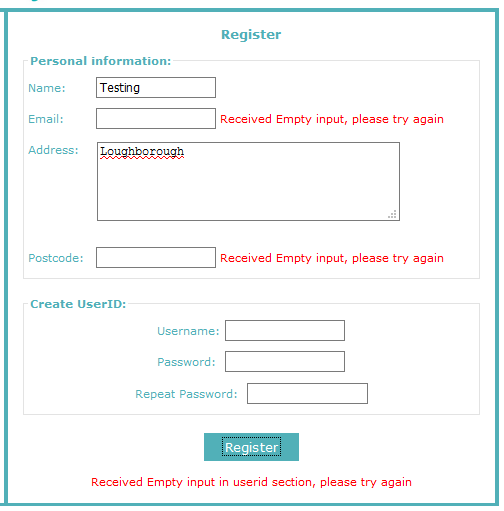
The main programming language used in this page is JavaScript, due to the large amount of validation that is being done. This is good because it allows the client to process most of the data on the page without having to interact with the server, resulting in a more streamlined experience for the user and less work for the server.

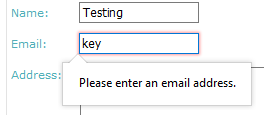
This page is fully functional and offloads as much work as possible to the client.

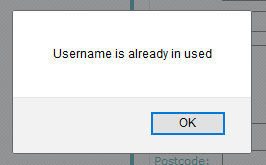
How to use the page

The user can access this page from any other page on the website. It allows existing members to log in to the website, whilst new users can create an account from the same page.

To register, fill in your personal information (your name, e-mail address, address and postcode) and create a new user ID (username and password), with the password repeated for confirmation, and press 'Register'. Ensure you have filled in all fields, otherwise an error message will be displayed.

The screenshot below on the left, shows the error message that will appear if one of the field are not filled in. If the Password and the Repeat Password are not identical to each other, an error message will appear on the screen and stop the form submission. This is shown on the screenshot on the top right. Finally the bottom right screenshot shows is the user enter an invalid email address, the form will not submit.

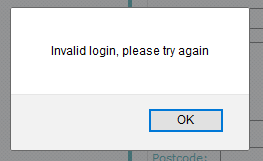
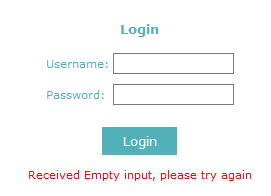




Also, if the inputted username exists in the database, an error message will appear stopping the form submission. The screenshot on the right shows this.

Once the form is completed successfully, your user ID will be added to the database and you will be able to log in from the same page. You will be logged in and redirected to the homepage automatically upon creating a user ID.

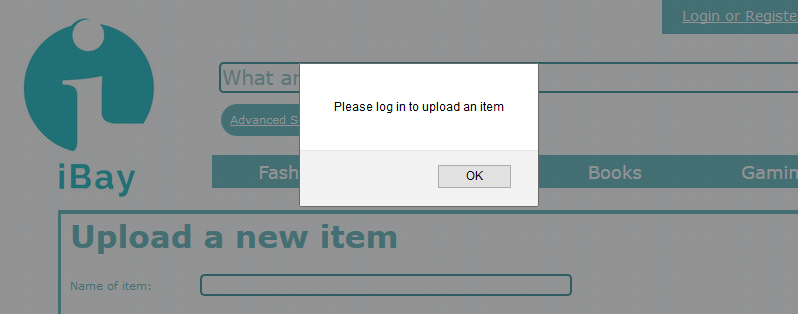
To log in, enter your user ID (username and password) into their respective fields, then press 'Login'. Failure to enter a valid ID will result in an error message being displayed (both fields required) .



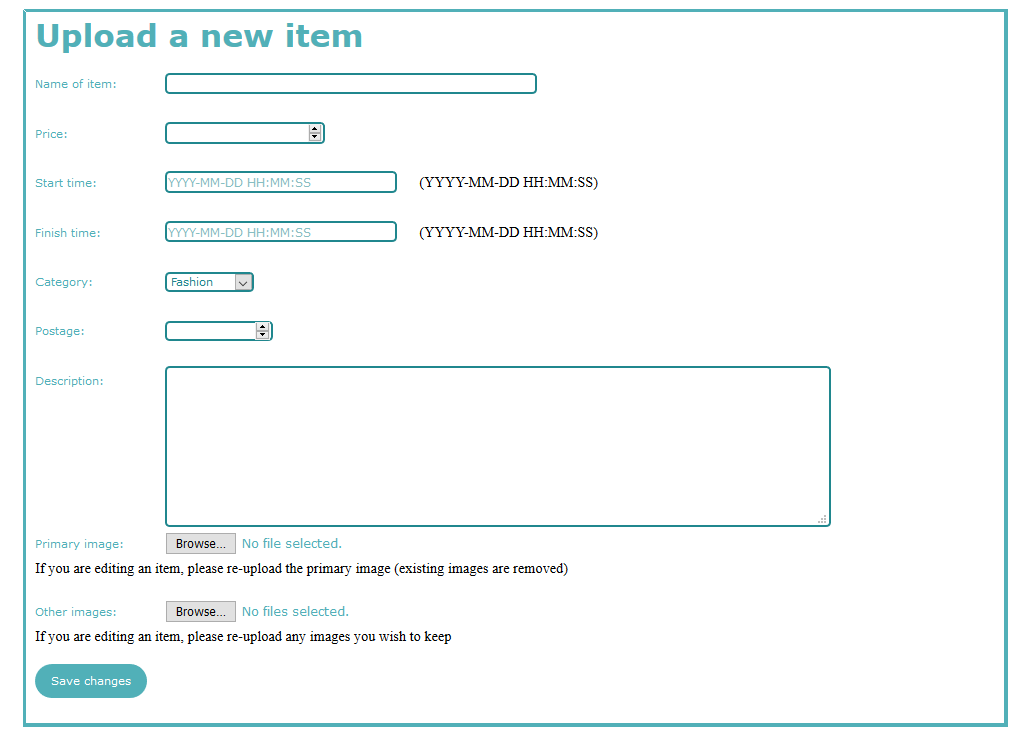
The screenshot on the left, shows the error message is both fields are empty.  
The screenshot on the right shows the error message when the user entered an invalid login. (When no match is find in the database with the given inputs)

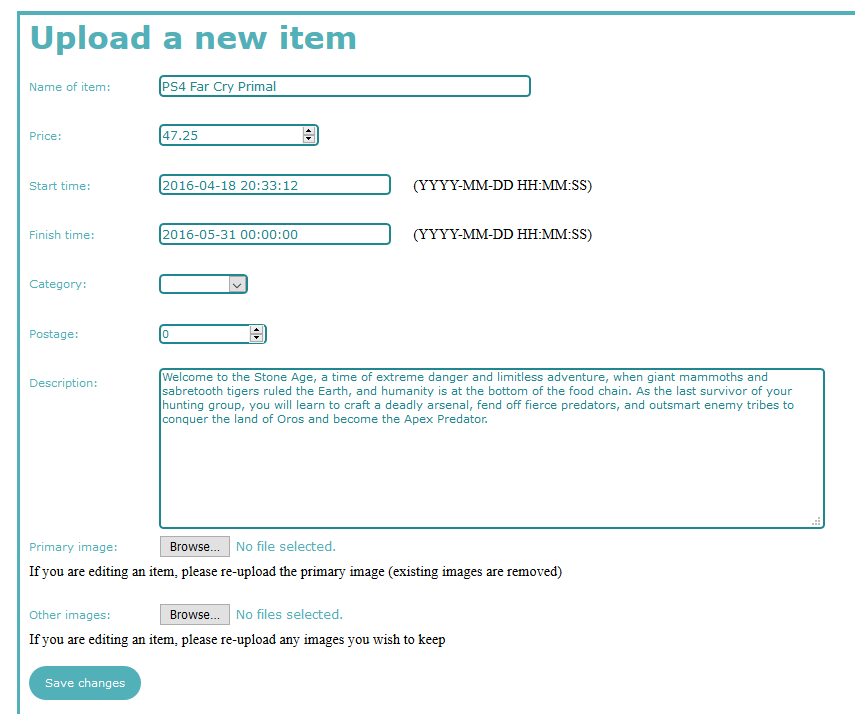
Once completed successfully, you will be logged in and redirected to the homepage. As a result of logging in, the 'Login/Register' button (displayed in the top-right corner of each page) will be replaced by two buttons: 'Your Account', which will take you to your user profile page, and 'Log Out', which signs you out of the system and ends the session.

**Upload**

The upload page allows the user to either upload an item to the database or update an item in the database.  
****

The user will be prompted to log in if they gain access to the page whilst not signed in to an account. They will be redirected to the login page after pressing the 'OK' button. The upload page can be accessed from the upload button or the edit item button in the userprofile page.

This is the version of the page a logged in user will see if they upload an item to the database. The form will initially contain blank fields, which will allow the user to enter the details of their item. All fields are required in order to submit the form.  


The user will see this version of the page if they update one of their existing items. The same form fields are present, each of which is required for submission, although the values found within the database are automatically filled in for the user. However, they will still have to re-select the images to be entered into the database.  


How the page it works

Upon loading, the method for accessing the page is checked. If the user reached the page without being logged in, they will be automatically taken to the login/register page of the website (using a JavaScript function). If the user is logged in, they can either upload a new item or edit an existing one, depending on how they reached the page.

If they are uploading a new item, a form will be generated (fields will not be filled in) allowing them to add a title, initial price, minimum bid, start time (i.e. when users can start bidding on the item), end time (when a bid can no longer be placed on the item), category, postage, description and images (at least 2, main image specified by user). All of these fields are required, preventing the user from submitting empty form fields.

If they are editing an item, the same form is generated, with the fields filled in using the item data from the database (using JSON and jQuery).

Once the form is submitted, the data from each field is extracted using PHP and prepared for the database. For the images, their binary (BLOB) data is extracted, as well as their MIME types and file sizes.

If the user is uploading an item, a MySQL INSERT query for inserting the item data into the 'iBayItems' table in the database is prepared and then executed. A separate query is prepared and executed for each image (via a loop), inserting them into the 'iBayImages' table in the database.

If the user is editing an item, a MySQL UPDATE query is prepared using the new data and then executed.

After the queries have been executed, the user is redirected to the user profile page.

How to use the page

To use the page, a user must be logged in. Otherwise, the page will automatically redirect to the login page.

Using the page is simple. Each of the form fields is clearly laid out. All fields are required for the form to be submitted, thus ensuring the user cannot leave any fields blank.

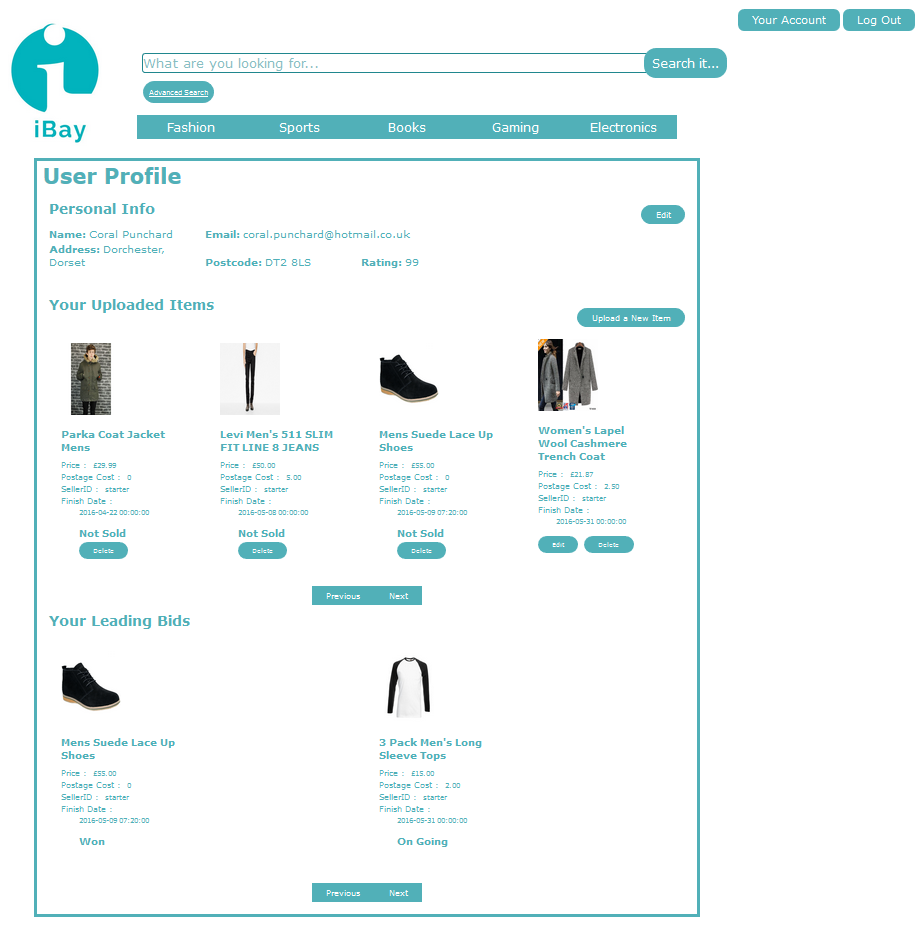
To upload an item, access the page via the user profile's 'Upload Item' button whilst logged in. This will load the page and generate a blank form. Fill in the form and upload the main image (to be displayed on other pages) and at least one other image for your item and press 'Save Changes'.

If all fields are valid, your item will be added to the 'iBayItems' table in the database and the images will be added to the 'iBayImages' table (with the same ID number in both tables).

To edit/update one of your existing items, go to the user profile page, find that item and press the 'Edit' button. You will be taken to an alternate version of the upload page with the current data (from the database) already filled in within the corresponding form fields. Make any desired changes and press the 'Save Changes' button. If all fields are filled in correctly, the item will be updated within the database.

After submitting the form (in either case), you will be redirected to the user profile page.

User Profile page



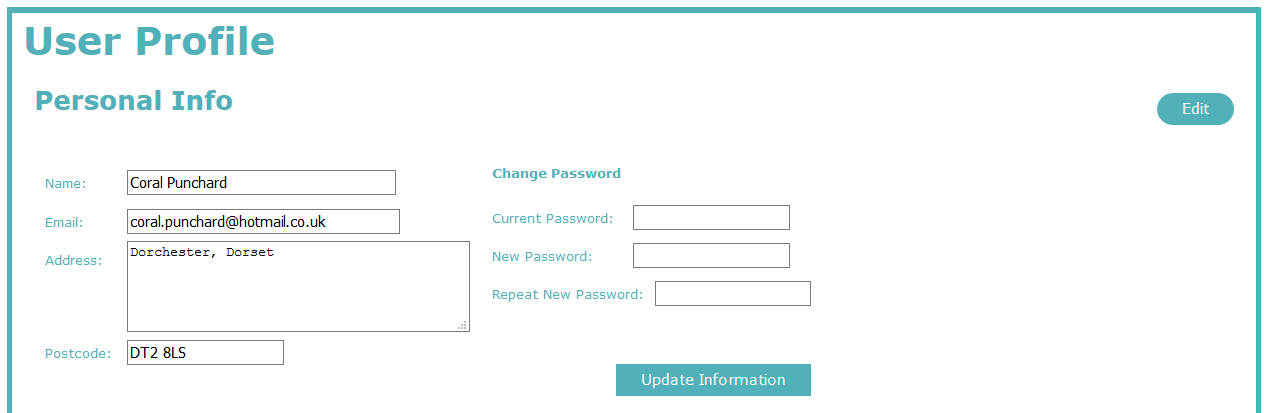
This is the header bar which is shown on the top of every page with the exclusion of the login page. This gives the sense of unity about the site having the same theme running throughout all the pages which the user will interact with

When the user clicks the edit button, it will change this section into text boxes so that the user may edit the information. The screenshot below shows this.

This button allows the user to upload a new item to sell. It will redirect the user to upload page.

Under this section, it allows the user to see all of their current items up for auction as well as the ability to edit or delete an item.

This section allows the user to view all of their leading bids on items they are winning on but also items they have been successful on and hence won the item in auction.



When the user is satisfied, they click the ‘update information’ button which will take the user back to the original page showing the new updated information.

The user can change their password if they wish, it is optional but the user **must enter the current password** to update the information.

This is what the page will look like once the edit button has been clicked.

How to use the page

When the user clicks on the ‘Your Account’ button shown once the user has logged in, they will be directed to the user profile page as shown in the image above.

In this page, the user will be able to view various pieces of information such as their provided information, their current ‘live’ items and what they have previously bidded on and/or won.

If the user needs to change any of their information, such as they’ve got a new email address, the user must click on the ‘edit’ button, located to the right hand side of the page in line with ‘Personal Information’. On doing this, the page will show text boxes in which the user can edit the information they provided on registration. Once this is completed, the user must then click on ‘Update Information’ in order for the information to be saved and updated. The user will then be re-directed back to the previous page where it will now display the update information provided by the user.

If the user would like to upload a new item for sale, they can do this from this page too. To do this, the user must click on the ‘Upload a New Item’ to which the user is then directed to the upload page. When this has been completed, the user will be re-directed back to the user profile page.

Within ‘Your Uploaded Items’ each image and it’s respective item name are hyperlinked, such that when pressed the user is directed to the Item Description page for that particular item, similar to when the user selects an item from the Search Results page. This is done by passing an item Id variable through the page using PHP.

Underneath each item is also a button to “Delete”. This, when pressed deletes the item from the database and also removes it from the User Profile page too. Again, this is a server side script which passes a DELETE query through to the database to make the correct changes.

Below this catalogue of images and information, there are two buttons, namely “Previous” and “Next”. These allows the user to shift the catalogue of images to left and right respectively, if there are more images to display in that direction. If not, a message is alerted to tell the user that they have reached the end of the images in that direction. This functionality is client side programming.

To the top-right corner of this catalogue of images and information sits a button labelled “Upload a New Item”. This allows the user, unsurprisingly, to upload a new item to the website and hence to their list of items being sold. Hence, this link redirects the user to the Upload page.

Below the Selling section exists the “Your Leading Bids” section. Listed here, in a catalogue of images and information of similar set-up and design to the “Your Uploaded Items” section, is all the items where the highest bidder of that item is the current user ie)the user can see which items they are currently the highest bidder on. The information displayed is the same as before and again the images and the item name links the user to that Item Description page. There is also a piece of text which informs the user as to whether the item has ended, and hence they have “Won”, or when the item is “On Going”.

How the page was created.

This was created in two sections focussing on the personal information and then the uploaded items and leading bids section.

Personal Information:

The main scripting language used here is JavaScript and hence mainly uses client side programming. The reason for this is because a function needs to be completed where the page is not reloaded hence using server-side here would not be applicable apart from where the information needs to be updated in the database.

Server side programming

Although the main section of this page uses client side, the main functionality of the update comes through PHP and mySQL query.

Before any scripting can be done, the database connection must be made. Through this, all the information from the iBayMembers table can be extracted using the mySQL code: SELECT \* FROM iBayMembers WHERE userId = ‘".$username."';"; This then retrieves all information being held in the database that is relevant to the logged in user.

After this, create an if(isset.. loop which will run if the update button has been clicked. Within this, there are two checks: Whether a new password has been entered and hence updates the database with all information including the new password, and then if this is not the case, it updates all the information to the database, overwriting the current information, excluding the password information.

**Client side programming**

Firstly, one needs to create a variable which can be used to manipulate the database information. To do this, PHP must be used within the JavaScript. Therefore, the command becomes:

var userinfo = <?php echo $result;?>; This then means that userinfo is storing all the data retrieved from the previous SQL query.

In the body of the HTML, using the variable name: html, create a long string of HTML code  through the paragraph tag <p> … </p> which displays the label of what the information will show, e.g. ‘Name’. Do this so all the fields are represented in which you want displayed: Name, Email, Address, Postcode and Rating. This paragraph gets the id of userdetails. To get the information to display, one uses ‘+userinfo[0].name+’. Here, ‘name’ represents the database column name in which the information is held, so to display the user’s address, you would use ‘+userinfo[0].address+’.

Then create a form with the id of personalInfo and then create a table where you enter HTML input tags to form text boxes. At the very end, create a input type submit in order to create the update button and close the table and form.

After this, create a JavaScript function which will check details. Within this, you need to assign three variables to the information given from the ‘personalInfo’ table. This information is the three password fields: current, new and repeat password. To do this, I used jQuery to get the information needed. The function will check if the current password matches the one given in the database, then it will check if the new and repeat passwords match in order to update the fields. Once this check has been completed, it will output an alert telling the user than the updates have been made.

By creating another JavaScript function called changeEdit() which uses the jQuery function .toggle() this can mean that the paragraph, userdetails and table, personalInfo can toggle between hidden and show which can be activated through the ‘edit’ button.

*Your Uploaded Items and Your Leading Bids*

This part there is only slightly more client side programming due to the functions needed however there is still a heavy basis of server side programming needed for the success of this part.

**Server side programming**

Initially we use server side scripting in PHP to get the $\_SESSION[‘username’] variable. This holds the username of the current user. Secondly we create a number of variables to hold strings of SQL commands that fetch all images and information from the database where the seller is the current user and also the images and information of all items where the current user is the highest bidder.

We then send query the database with each of these variables and store the results in PHP arrays if results are returned. We check for errors that the database returns, and if any occur, kill the request and get the error message.

Finally, on the server side, we use json\_enode to encode the PHP arrays of results such that we can then use these arrays within the client side scripting. This is done for the reasons stated on when to use client side.

**Client side programming**

We start by setting variables in JavaScript equal to the encoded arrays in the PHP code. We also set up other variables that store things such as the number of items the user is selling and the number that they are the highest bidder of.

We then use JQuery’s $(document).ready() function to set up a function that will run on page load.

We create variables to store HTML code, intersected with the JavaScript variables already declared, and then use JQuery to append these strings of HTML code to the relevant location in the pure HTML script.

We declare JavaScript functions to accomplish tasks client side. One function sets up a table that will store images and their respective information from data which is passed into the function as input. The function is called to set up both catalogues of selling and buying items.

We also have JavaScript functions that change the images which are visible. Each of these functions uses validation to check whether there are excess images to show or not, and then calls the createtable() function described above to create the image and information catalogue.

At the bottom of the code, we use purely HTML. We keep the organisation very clean, as most of the work has been done within the client side functions and the JQuery. Hence, the HTML purely sets up the tables and divs that will store all the information that will be appended to them within the client side scripting.

We also use this section to set up forms. We use forms to pass hidden variables through to the next page when the user presses a button. For example, when the user presses the delete button on one of their items a client side function sets a hidden variable within the ‘hiddenDeleteItem’ form and then submits the form. The form will then redirect to itself, but then server side scripting will have a variable passed through.

Hence the page uses a variety of server side and client side programming. However, we have tried hard to ensure that server side has solely been used for the parts that needed use of server side programming. Hence, it is only used for accessing session variables, variables passed through from previous pages via forms and to fetch data from the database.

We use client side for everything else, including validation and HTML code creation.

Other Pages

dbconnect.php

This file contain the main information for setting up the database connection whenever we need to query, insert, update or delete from the database. The php function: ‘include “dbconnect.php”’ is used to include this file in the other webpages.

SearchQuery.php

This file is created solely for sql queries where we need to get the images from the database. The file request a sql query and run the query in the database. Then, all the images obtained from this query will be converted into base64 before we return the query results as a JSON array object. This file is mainly used for the AJAX $.get() function in the other webpages.

**Conclusion**

On completion of the creation and implementation of the website, the group has agreed upon a number of improvements that, given more time and research into the methods of implementation, we would have liked to introduce into our website.

Firstly, we would have liked to add more pages/functions. For example, having an alert function to each user, possibly by an internal website account message or an external email sent to the user, if they have successfully sold, or successfully bought an item. Further to this, adding a user feedback system to that this would allowed sellers to rate the buyer on things like communication and speed of payment, while the buyer could rate the seller on things such as speed of arrival of the product.

Moreover, we would have also liked to have implemented more advanced concepts into our website code. For example, the use of animation for a more pleasing visual effect on the website. Furthermore, although the task did not require it, it would have been interesting to research how to ensure security within the website on parts such as password validation and setting new passwords as well as in our server side programming to ensure that no malicious software/users could hack it, due to the reasons mentioned in the disadvantages of server side and client side programming.

Finally, it would have been ideal to agree upon a coding style that all members of the team could have tried to stick with. For example, naming of variables. This would have made it slightly easier for collaboration on each page and to understand each other’s code to assist when problems occurred. However, in a group of individuals this was inevitable.

In conclusion, although we have listed above some possible changes that we would have liked to have implemented, given the time and our initial lack of knowledge of the programming languages used within this project, we all agree wholeheartedly that we have enjoyed and benefitted from this Team Project. We have learnt a lot, both in terms of technical knowledge of languages and applications as well as soft skills such as collaboration, aiding others when they face problems and deadline keeping. We believe the website works very well and has good overall aesthetics. Every member is satisfied that we were thorough in our attempt to ensure the use of client side and server side scripting was used for the most efficient compromise.