# Web Application

## Requirements and design

The design of the application was decided very early on in the development cycle and remained fairly consistent throughout the whole life cycle but there have been a few notable changes made.

### Design Changes

One of these most notable changes was the change of colour scheme. The colour scheme at the first usability study was blue, black and white. During the study it was commented that the entire application was quite dull and should be made to feel and look more engaging and so the team opted for the current colour scheme of varying shades of blue, green, white and a little bit of orange.

The member’s home screen also went through a change which removed the member’s details as this is displayed on their account page. In the place of this was put notifications about statuses of the member’s bids and transaction. These notifications show the number of bids or transactions affected and only show up if there is a change such as one of the members’ sent bids has been accepted or the member has a bid sent to them which they need to accept or reject.

All the pages which show advert also changed from the initial design. At first the adverts where displayed in table format with each advert taking up a row of the table. This was altered so that an advert would be a larger white rectangle consisting of many rows showing information about the advert. The title of the advert was larger and bolder while other details were not so prominent. One major improvement was the addition of an image all adverts. If a member does not include an image then the system automatically uploads an image relating to the category of the advert.

Another feature which has been added was a tabbed feature for the bids and transactions page. At first the different type of bids and transactions tables were placed one below the other but after using the system it was realised that once a member accrues a large number of these then they would need to scroll a long way down to reach the second or third or even fourth table of bids or transactions. Having the tabs at the top for each type of bid or transaction allows for much quicker access to the required table. One handy feature of this was that when a notification was displayed on the member’s home page then clicking the notification would take them directly to the relevant table. For example if a member sent a bid which has now been accepted then clicking on the notification would take them to the tab on the “My Bids” page which shows the accepted bids.

The ability to search other members was added very late in the development. A member can be searched by either entering their forename, surname or both. It was decided that all members should be able to see other members’ names and balances as well as any reviews given to them. The decision to do was due to the fact that a LETS system promotes openness about other members and ensures that members do not accrue very high balances without giving back.

Icons have been added to almost every action to promote dual coding of functionality. Examples of this are the home page link has a house next to it and the help page contains a question mark enclosed in a circle.

### Features not included

Overall the application has included many more features than was originally intended but there are some features that the team wanted to include but didn’t due to time constraints were:

* The ability for a member to edit an advert already published.
* Pagination on adverts – Currently if there were 300 adverts in the system then all 300 would be shown on one page. It would be good to be able to navigate about 30 or so on one page at any time.
* View transaction statues on the same page as the advert they belong to. Currently only the bids of an advert are shown on the same page as the advert.
* A messaging service for members to communicate with one another to make the process of organising selling and buying easier.
* A forum where members could post discussion topics and receive help about anything they wish.
* Allow for a member to reset their password if they had forgotten it. Currently they can change their password only when logged in or contact admin to change it if they have forgotten it.
* The ability to filter adverts by date added, expiry date, category, advert type (Offer / Request) or item type (Product / Service).
* The ability to search for adverts posted by a specific member.

## Software Engineering

Designing a web application using JSP and servlets was a first for the team. The first task was to get familiar with the workings of this technology. Due to our limited understanding the initial structure of the web application was not as well formed as it is in the final release.

Initially all JSP pages were treated like static HTML pages and each one contained duplicate code such as headers and sidebars. Over time the team learnt that large chunks of code could be placed into separate JSP files and then included using the include tags provided by JSP syntax. This was hugely beneficial for further development as it encouraged DRY “Do not Repeat Yourself” coding. As an example the sidebar was used on about 12 pages and consisted of about 80 lines of code. Now this code was within one file and as can be seen would obviously reduce code repetition by a large amount. It also meant that when one link’s URL changed then the change could be made in a single file rather than in all 12 pages, reducing the chance of errors.

With this knowledge a lot more code was taken out and put into separate JSP files such as tables of bids or adverts which are exactly the same but in different pages. This way of structuring the code was a lot better than repeated code in many pages but as it turned out was not the best approach we could take. Code that called web services was being placed in JSP files and this was very difficult to debug and control errors. The next step taken was to take this code and put it into servlets. What this now made possible was to call all required web services before the page has loaded and if an error occurs then catch that error and notify the user of the error on the page after loading. Using servlets also allowed for multi-threaded code. Whenever multiple web services were called such as getting 4 tables filled with different types of transactions then each of the 4 web service call were run in their own threads. The effect this had on page loading times was very significant. As an example, the ‘My Transactions’ page would normally take 9 seconds to load but due to the multi-threaded code it now takes about 1 second, obviously depending on the amount of bids the member has.

Another feature worth mentioning is the filter used on almost all of the pages in the application. This filter is run every time a page is requested. If the page is only allowed to be used when a member is logged in then the filter will activate and check if a member is actually logged in thus preventing just anyone accessing these pages directly.

Classes used within the application have been placed into packages according to their functionality to make it easier to understand for future developers as well as easier to maintain. One such class is the web service connection class which has its own package and could be reused by many other applications if so desired.

The use of a data model with the web application was very different to how the desktop application used it. The entity classes such as member and advert where only used to hold data and then the object converted to JSON to be transported to the web service. There is no relationship between the classes. However, the data model has been placed into a library which both the API and the web app use. This meant that if the data model needed to be changed then it would only be changed in one place and not in both the API and the web app.

### Problems

One problem we have had was that once a servlet is called which calls a web service to insert a new entry into the database then the page refresh button could be used to reinsert the same data. This could result in quite a lot of problems for members so the way the servlet redirects to other pages needed to be changed so that pressing the refresh button on a page did not resend the new data to the database.

The application is very stable and there are not many times were an error happens but when such an error does occur there has been code put in place which tries to catch these errors and redirect the user to an error page from which they can login again. This however does not always work and in such cases the web page just goes blank/white and the user needs to open a new page manually.

In order to increase page load times, the member’s details of who is logged in is saved into a session variable. This works perfectly for most of the time except in instance when the members details have been changed by another source such as they have been paid money or the member logged into the Android application at the same time and changed their details. To overcome this would mean to request or check to see if the details have changed recently but this approached takes away the benefit of saving the details in a session variable.

### Conclusion

Learning to program JSP and servlets has been a challenging but welcomed task as a lot of new skills have been learnt. Future developments of a similar nature would undoubtedly be much more seamless due to the knowledge gained by undertaking this project.