CO600 Final Year Group Project

Javaplets

A Portlet production project

Name, Affiliation, Email address

Abstract

This paper presents the technical details of the Javaplets project. The purpose of the project was to create several new Portlets for the existing University of Kent portal system. This paper will look at the aims, design, implementation and testing of the Portlets. There will be a summary of the tests used on the Portlets and details on how the Portlets were created and the techniques used to make them function correctly. The outcomes and the overall successes and failures of the project will also be analyzed, paying particular attention to the problems encountered throughout the project.

Introduction

In a society where people have less free time, the demand for convenience is a higher priority than ever before. People feel the need to have everything in one place, especially when it comes to the web. In this case it relates to the University of Kent students and the tools that they require in every day studying. A current partial solution exists in the form of the student portal. This allows a student to view their emails, connect to the library and view webCT, which is a facility where lecturers can add additional learning material for students to access as they please. Feedback from the existing portal found that students believed that some parts were missing. For example, the general feedback showed that some aspects such exam timetables were not personalized therefore it took time to locate their modules. An example solution for this problem would be to create a personalized exam timetable for each student. The premise of this would be the student would log onto the portal and from your user name, would be able to take the modules you are currently taking and retrieve an exam timetable specific for that user. We intended to develop this need for personalization by creating Portlets to aid a student in their busy schedule. Although study is an important part of any students life, so is the social side that university is well known for. Therefore it is important to communicate with other students, whether it be work related or not and find a way to shut off from the stresses of deadlines. The solution to this was looking into games that students can play within the portal and different means of communication. The solutions, known as Portlets, were created using software called 'Net beans', which permits the use of a combination of java and xhtml to produce the desired effect.
We have created Portlets, which can be thought of as containers within a bigger container, known as a portal. Each Portlet is independent from another and displays different data in a meaningful and dynamic way. Portlets can generate Markup fragments, although you cannot send someone a URL to a Portlet as they are not URL addressable. This was not a problem in the context we used it in as a student will only need to be directed to the University of Kent portal page to access the Portlets. For the purposes of the project, the Portlets were implemented in a portal system called uPortal. We used this as it is very similar to the existing University of Kent portal system.

Background

The motivation for this project was to work on the current portal system. The existing portal was made up of mainly links, which allowed users to branch off to different pages such as emails. As previously mentioned, the feedback from students felt like there could be more steps taken to make the portal more complete, making it so users do not need to leave the portal to complete tasks. It was felt that the portal was missing features that would be useful for any student, and so talks began to discuss which Portlets would be the most suitable. Firstly, the decision was made that five Portlets would be a suitable number to implement. The skill level was also important as it was necessary to get each Portlet to challenge a new aspect of programming, whether it is using RSS feeds to retrieve data or creating connections using sockets. It was clear that time was going to play a large factor in the decision making criteria of which Portlets to create. Once the decision had been made on which Portlets to create, a plan had to be created. Due to not having any previous experience on Portlet production, the plan was only a rough guide that would need updating and developing as time progressed. The plan we used a simplified Gantt chart as we felt that the project was not big enough to create a full Gantt chart. The first step in creating the plan was working out how many weeks we had in total to complete the project. We also took into account holidays, for example the Christmas break and reading weeks. From here were created the Gantt chart making assumptions about how long we thought each Portlet would take. For each Portlet we assigned a head programmer with an assistant programmer, so paired programming could be achieved. The remaining group member would be in charge of completing the necessary documentation and would also play a part in the testing of the Portlet alongside one of the programmers. We felt that this would be the most efficient way of developing knowledge of Portlet production and the fairest way to

split the heavy workload. The plan was updated several times as some of our assumptions were incorrect and problems in communication with some members in the computing department meant that we could not progress as smoothly as we would have liked.

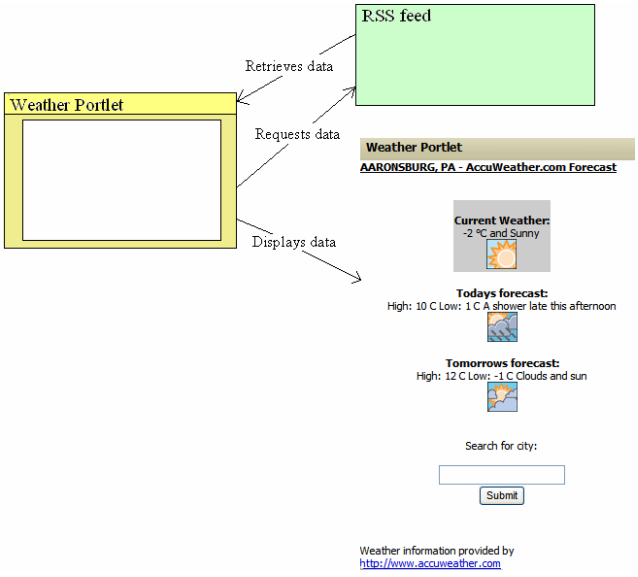
Aims

The group's main aim was to create five working Portlets that possibly could be integrated into the existing University of Kent portal. The Portlets that were decided upon to be created are as follows:

- A weather Portlet, which would allow the user to view the weather for any city around the world.
- A forum Portlet, which builds on the existing news groups to create a clearer and easier to use system showing only the module forums that they are currently taking.
- A reading list Portlet, which will allow the user to view the reading list for any module that they are taking, with the added functionality of being able to view any module reading list the user chooses.
- A game Portlet, which involves creating a version of the Battleships game, allowing users to battle opponents on or off campus.
- An IRC Portlet, which lets computer science students communicate with each other in a chat room based system.

Weather Portlet

The first step was to figure out what the Portlet was going to do. It was first decided that there was going to be a set list of locations that you could select, for example, the major towns and cities within the United Kingdom. We then decided that we would try and use an RSS feed to allow the user to manually enter any location in the world and view the weather, therefore increasing the scope of the Portlet significantly.
However, because this was the first Portlet that we had ever created, it took a while to figure out how everything worked. We created a shell script to handle deploying the Portlet code. The first problem we had was that we planned to use 'Eclipse' when creating the Portlet; however, due to access rights we could not download the necessary plug-in. The result of this was to use Netbeans however we could not get Netbeans to work with uPortal, so we made sure that the script conformed to the jsr168 standard, a standard necessary for writing Portlets.
The first RSS feed that we intended to use was courtesy of channel 4, however initially we struggled to get the feed to work. We discovered that the whole document was being passed, however all that needed to be done was to pass the document via a URL, which solved the problem. Another problem we had with the channel 4 RSS feed was that it had no visual displays for the weather; therefore we could only display the temperature. This was deemed unsuitable as the user would not be able to see what the weather was, defeating the whole object of our Portlet specification. We then began a search for another RSS feed that would be able to provide images associated with the current weather. We found a more suitable feed from AccuWeather.com, which we wanted to use. Reading the terms and conditions for using the RSS feed, we saw that the feed could only be used for private use. We emailed AccuWeather.com outlining the project we were undertaking and requesting permission to use the feed, which they kindly accepted, as long as we sent them screen shots of the finished Portlet.
Using the RSS feed, we were able to extract the data that was useful and displayed this is an easy to read, easy to use Portlet.
Below is a simple diagram showing how the weather Portlet displays the data.

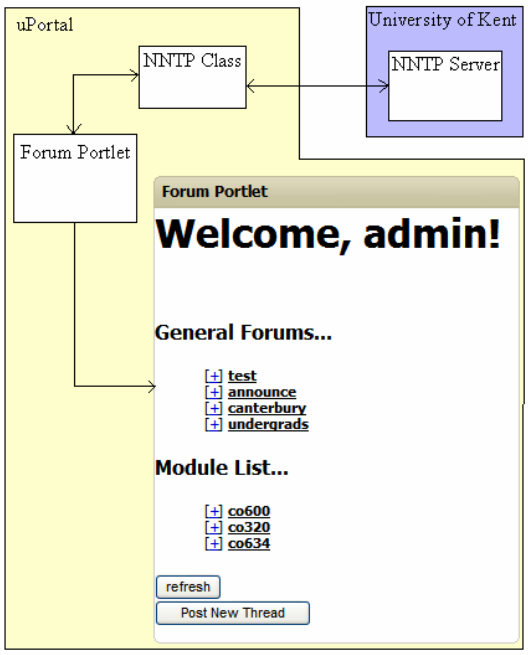


All of the information that we displayed is from the RSS feed. All the Portlet does is work out what

location is to be displayed if a new one is entered and actually parse the RSS feed to display the wanted information.
During testing, it was found that several bugs were present in the code which meant that it was not ready for release. The main problems were that invalid or null data was not handled correctly, resulting in the system crashing. If a user entered an invalid location, there was no option for the user to re-enter the location. We changed this by simply adding an input box that accompanied the error message.
User evaluations showed that people were happy with the outcome of the Portlet. The main area of positive feedback was the result of the amount of different locations you could access. This was all thanks to the RSS feed that we managed to get from AccuWeather.com, which was the basis of this Portlet.

Forum Portlet

The specification for the forum Portlet consisted of making sure that students were assigned to the forums for the modules that they were currently taking. Also, we decided that the user should have several default forums, which were forums that we thought would be useful for any student. These included 'announce' and 'undergrads', which are forums used for general notifications. The functionality of this Portlet is based on existing news groups, however developing functionality such as replying, nesting, editing and deleting posts.
However, when it came to creating the Portlet, we had a severe communication problem. Initially we did not know the best way to communicate with the forum, so we sent emails to various members of the computing team. The result was that we were passed from person to person which slowed down progress greatly.
Another problem that we had was that we displayed unnecessary data, for example, when we displayed a post, 'null' appeared at the top. To solve this problem we had to start displaying after the 4th character, therefore ignoring 'null'.
The second problem we had with display was that when a user had selected a module and could see all the posts, when they tried to minimize the list of titles, they would only be hidden and therefore a large gap would appear between the module headings. This gap varied depending on how many thread titles there were. The solution to this was to use JavaScript to remove the subjects as opposed to hiding them.
Below is a simple diagram of how the forum Portlet works.



The forum Portlet works by initially sending the users log in name to the NNTP class. This class then sends the log in name to the NNTP server, which retrieves the user's modules and sends back the appropriate information. The NNTP class then sends this to forum Portlet, where is it process and displayed as shown above.
When testing we found that the Portlet would time out if there were too many replies. To solve this problem we made it so the threads were only retrieved when the list of threads was expanded. This meant that threads were not unnecessarily being returned and this reduced the workload of the Portlet.
After overcoming these and several other problems, we felt like the forum Portlet worked as stated in the specification. A user could reply to a thread, create a new one, delete or edit their own posts, view new posts and see how any replies were made to each thread.
After completing this Portlet we got students to use the Portlet to see if anything was missing, incomplete or not necessary. It was found that students were very happy with both the look and feel of the Portlet and felt it would be useful addition to the student portal.

Game Portlet

When writing the 'Project Plan', and deciding which Portlets to do when, we can upon the realization that any worthwhile game implementing Portlet would have to be complex enough to allow more than one user to interact with each other. As such, the notion of a

multiplayer game was discussed and Battleships were chosen as a suitable game to try and implement. This was chosen, as the overall nature of the game is pretty simple and straightforward, meaning that we could spend more time dealing with the more complex task of getting at least two users communicating with each other across the uPortal system.

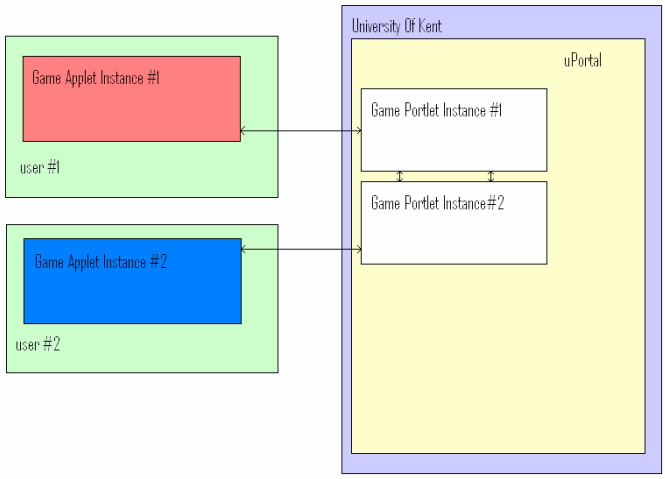
During the initial planning of this Portlet, it was already clear what sort of potential problems we might come across. User interface and communication were the main hurdles we had to try and overcome in order to end up with a Portlet capable of producing behaviour comparable to that of a traditional Battleships game.

We felt that this type of game would offer us the more challenges of all the Portlets we had managed to think of, as well as enabling us to come up with new implementation ideas and structures that could be built upon during later Portlet implementations.

Once the go-ahead for this Portlet was given, it was time to research exactly how we could implement a battleships game within the uPortal framework. After some initial research into how uPortal behaves when it comes to clients and servers, we came to the realization that the Game Portlet instance could be used as a server between multiple players. This meant that actual communication between users was done locally. This would enable both on and off campus users to use this game whilst keeping communications across internet connections fairly low. The main problem with this was that, as far as we could tell from our research, whilst Java Sockets could work within the uPortal system, numerous problems may occur when trying to distinguish one user from another. A solution to that, which we decided to implement, was to have a class dealing with user information. This class would get the user name used to log into the uPortal System, and use it as an ID of sorts so as to keep other data relating to that user as the game progresses.

Further research into having a game Portlet, led us to conclude that users would be more interested in using the application, should it be able to deal with them parting the game halfway through and being able to come back later to finish it off. This means that user could have a round or two and be able to log off and return later to continue, with the Portlet storing their ship positions, number of shots fired and where, and current score. This would mean that user do not have to finish the game when they start it, adding to the appeal of using it and enabling them to be able to come back to it after weeks of inactivity and continue where they left off.

One of the most important parts when dealing with Portlets like this one is the designing of its interface. We wanted to keep the overall look of each of our Portlets as small and simple as possible, as that is one of the appeals of Portlets, so we eventually settled on a design that incorporated as much as it could, without any repeated elements or looking too cluttered. The display grid would be used to display both the user's own ship data and opponent shots whilst being able to switch to their opponents grid for them to have their go. We also decided to include a chat system, so that both users could communicate with each other without having to use another communication application.



The implementation of this Portlet was really three fold. Firstly, a useable interface needed to be created that could be updated in the form needed, i.e. being able to have a grid switch between the users data and their opponents data, whilst still having colour and graphical images. The uPortal System mainly uses java behind the scenes, but to actually display information, only HTML, JavaScript and XML can really be used. As such, our conclusion for this was to make the interface as a Java Applet that would communicate with the Portlet displaying it, whilst the Portlet would act our main server for all the game mechanics.

Secondly, to reduce 'lag time' between players moves, the server would have to deal with as little information as possible. This meant that the data surrounding the game mechanics had to be investigated and categorized into what the Java Applet interface could deal with, e.g. loading of graphics, and what the server would need to know, e.g. ship locations on a user's grid, enabling us to only have to notify the server of changes to coordinates if a shot has been fired.

Lastly the chat system had to be implemented to enable communication between users. This turned out to be quite an interesting idea to add to the game Portlet, as it enabled us to be able to give more information to both users from the server. In essence, the chat system would display communication between both users and any information the server needed to convey across, reducing the need for dialog boxes.

Whilst we had a good idea of what needed to be done for this part of the project to be successful, it was not without its fair share of problems. To enable the server to distinguish between users, and to make sure the communication between the Java Applet interface and the Portlet server was as secure as we could make it, some form of authentication needed to be in place. The Applet would connect to the server via a socket connection and send across an authentication byte that would then be checked by the server to make sure it is a valid one, then linked to their username to be able to distinguish between connections. This, however, lead to quite a large problem. Whilst the applet connected to the server, it only did so once out of 6 attempts and then crashed the uPortal system on the 6th try at connecting. Due to this problem, Portlet production speed was reduced dramatically as we attempted to figure out why this was happening. Eventually we had to move on to the next Portlet, the READING LIST Portlet, in the hopes that working on a different Portlet might give us enough change so that when we'd come back to the GAME PORTLET, we'd have a fresher view of it and maybe fix it. Unfortunately this did not happen, we therefore continued onto the next Portlet, the IRC PORTLET. During the implementation of this Portlet, the similarities between this one and the GAME PORTLET connection and authentication became apparent. It was decided, that the connection/authentication would be rewritten from scratch for the IRC PORTLET in the hopes that a working method would be found. Thankfully it became apparent that the order of events leading to the connection and authentication needed to be in a certain order for it to work every time. Upon realization of this, the GAME PORTLET connection and authentication method was rewritten to reflect this and was successful in making it connect every time. With this major problem solved, production of the GAME PORTLET resumed and soon we had a working Battleship game. A 'lobby' system was added to enable users who enable the Portlet to be able to view a list of games awaiting a player as well as being able to view a list of games they are currently playing and have not finished. The 'lobby' system was implemented to try and give users a chance to interact with anyone logged into the Student Portal System, they could even play total strangers and chat to them.

Reading List Portlet

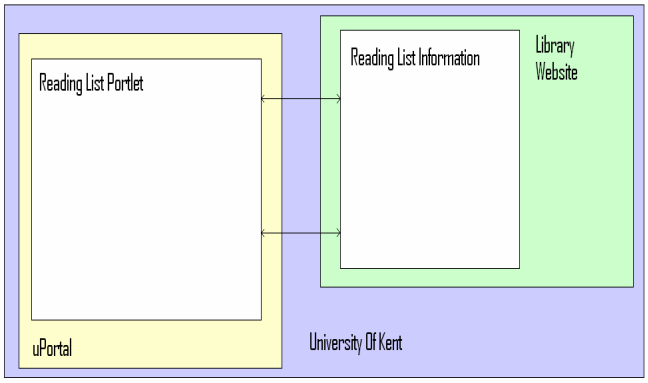
At the start of this project we were given the opportunity to look over information gathered from feedback samples of the University Of Kent's new Student Portal System and later use and, to some extent, play around with a beta build of it. From these were able to collect various ideas for our list of Portlets. Whilst looking over this information, it became apparent that, of the users that had used it and left feedback, some way of check Reading Lists for modules would be quite a welcome addition to the current list of Portlets used.

Upon researching what current system users had at their disposal to check for any reading list for a module, we came across quite an extensive collection within the University Of Kent Library website. This contained Reading List information for almost all subjects and departments, meaning that if we were to use this as a source for our information, this Portlet could be used by any potential user from any department.

Further investigation into the current Reading List system, revealed that whilst the information is readily available, little web services exist [at least, at the time of our implementation] to access this information from another 3rd party application. Contact was made with one Ben Charlton and a meeting established to talk about what the University Of Kent would be doing to the Library website over the coming months and whether there would be any services we would be able to use within the constraints of the CO600 Project timeframe. Unfortunately none existed that we would be able to use resulting in a technique known as HTML scraping having to be used to get the data we'd need to display.

Once we had an idea as to how this Portlet would get its information, a design had to be chosen that would be easy to use and as simple as possible. The actual look and feel of the Portlet was fairly straight forwards.

As with the FORUM PORTLET we assumed that, should this Portlet be used by the University Of Kent, the web service to retrieve a user's registered list of modules would be used, and built our Portlet around that premise. After we had managed to get a fix list of modules to be displayed with access to the corresponding Reading List, it became apparent that users might also wish to check reading lists for



modules other than those they were registered for, i.e. if they wanted to check out the list of books for optional modules they would have at a later date. A search box of sorts was added, in which a user could just type in the module code and the Portlet would retrieve the Reading List for it, essentially giving users another method of accessing the Reading Lists.

Displaying the data from the Reading Lists was also a challenge. For some modules, the Reading Lists are quite extensive, displaying numerous book titles. Whilst all of these might be considered useful for the user to know, a choice had to be made to keep the Portlet relatively small and portable. In the end, it turned out the Reading Lists had an 'Essential Reading' element that we could use our HTML scraping method on, letting us display within the Portlet only the bare minimum information that they would need. Not fully satisfied with this, the usability of the Portlet was extended to include links for each book entry so that users would only have to click on a link positioned next to the book entry where a new Internet Browser Window would open and display the Library Catalogue entry for that book. A specific link to the module Reading List Webpage was also added to make it easier still for users to be able to view the complete reading list.

IRC (Internet Relay Chat) Portlet

Our final Portlet was chosen to be an IRC [internet relay chat] Portlet in the hopes of introducing more students to the University of Kent IRC network. Whilst by this time we had familiarized ourselves fairly well with the Portlet production procedure, we were getting closer to the project deadline. This factor coupled with the intricacies of an IRC client meant that implementing our own client was out of the question if we wanted to hand in a finished and working application.

Following some research, the Java Applet PJIRC came to our attention. It was chosen to be the heart of the IRC Portlet due to our previous experience with Java Applets. The main purpose of the actual Portlet container in this case is to point the PJIRC client to the correct ports, authenticating each connection to it so that it can associate users with connections and enabling off campus users to still be able to access the University of Kent IRC network without having to be on campus.

The implementation of this Portlet was quicker than previously thought, as similar code was used with the GAME PORTLET enabling us to be able to reuse and modify the majority of the code needed for the interaction between Java Applet and Portlet interaction.

Essentially, this makes the IRC PORTLET nothing more than a proxy between the PJIRC Java Applet and the University of Kent IRC network. This does mean that most Java Applets should be easily adaptable for use with a Portlet system similar to uPortal.

Conclusion

Our aim for this project was to produce 5 fully working and tested portlets with full documentation that could be integrated within the new Student Portal System at the University of Kent. What we have managed to produce is 5 fully working and almost completely tested working portlets. It must be said that due to forces outside of our control, parts of this project were slowed down and took longer than anticipated to start up again. Unfortunately the IRC and GAME PORTLETS are not as tested as we would like them to be, although they are fully working to the best of our knowledge. All portlets created are functional and can be used within the new University of Kent Student Portal System with little to no modifications needed as we made sure the uPortal framework used to run these was as close to the university's as we could make it.

Our portlets are at least comparable, if not as close to, other similar portlets used by the University as we could make them given the resources at our disposal. That being said, there is still room for improvements or further features to be added to them should others choose to do so. With this in mind, we have attempted to document each Portlet production as thoroughly as possible with as much detail as possible.

It is our belief that most of the portlets we have implemented, if not all of them, would add further customisation to the University's new Student Portal System. Enabling more students to access a wider range of services whilst, hopefully, still providing them with a worthwhile range of non-academic oriented portlets at their disposal.

Acknowledgements

We would like to acknowledge and thank the following people, in no particular order, for their time help and input with our implementation of these portlets:

- Bonnie Ferguson – for her interest in our portlets
- Phill Camp - for his help with the Forum Portlet
- Ben Charlton – for his help with the Reading List Portlet
- All the UKC students that helped with our testing
- AccuWeather.com – for letting us use their RSS feeds
- Anyone else we may have missed,

And finally:

- Peter Rodgers – for supervising and providing us with advice in relation to this project

Bibliography

PJIRC – www.pjirc.it

AccuWeather – www.AccuWeather.com

'Building Portals with the Java Portlet API' by David Minter and Jeff Linwood

uPortal Home page – www.uportal.org

uPortal Wiki - www.ja-sig.org/wiki/display/JSG/Home

APPENDICIES

CO600 Group Project - Javaplets

Project Plan and related changes

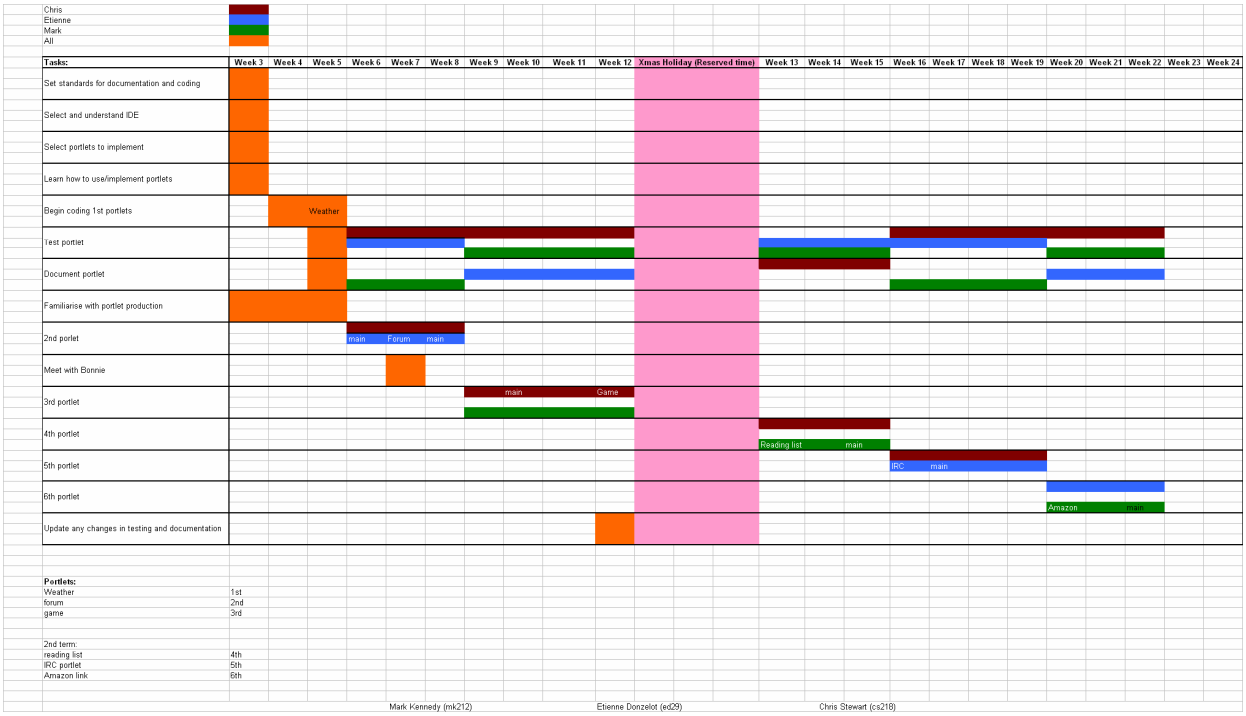


Figure 1 – Original Plan, created in week 1 of Project

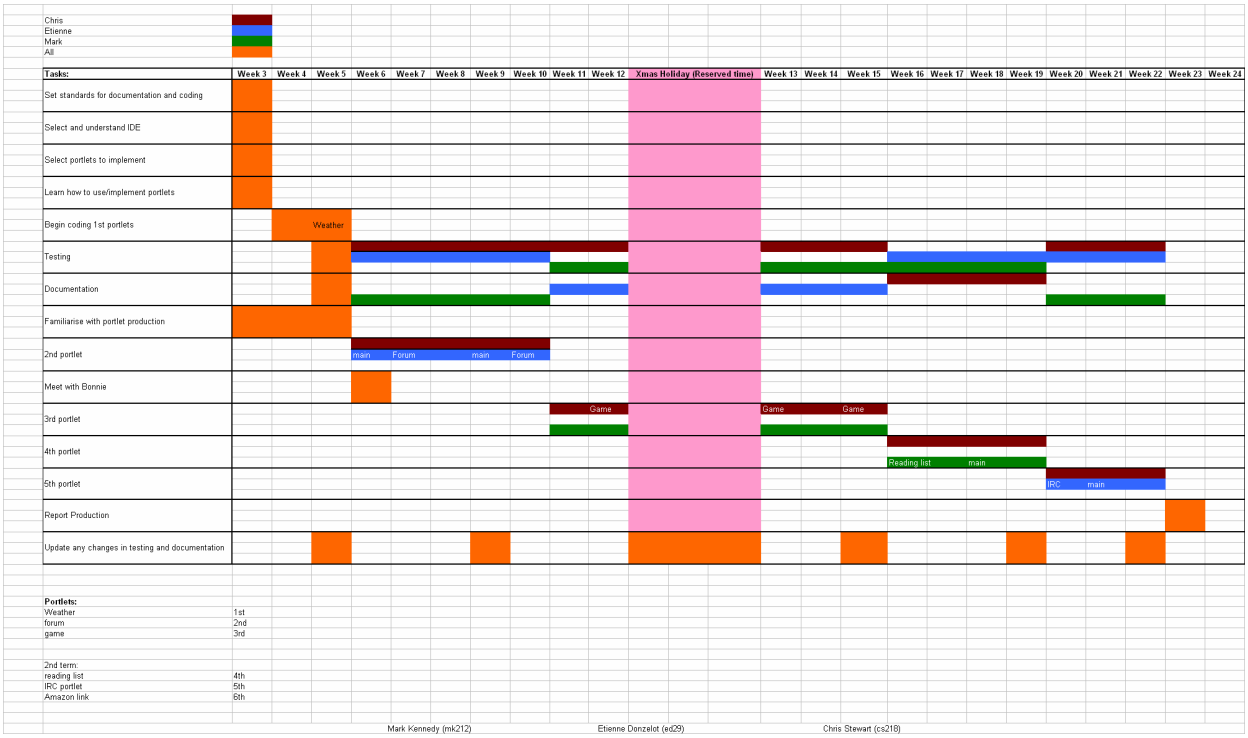


Figure 2 – Revised Plan, last edited in week 20 of Project

Whilst our original plan was not a bad one, it became clear to us that we had been a bit over ambitious with some of the time allocations. Our first portlet production did stick to the plan, but early within the second portlet production we came across various problems which slowed us down. From then on, it was decided that the plan would be checked at the start of every week and updated if needed. What we had failed to take into account when first making the plan was that University inter-departmental communication would be slower than we expected. This meant that what we had hoped to only take a couple of days to a week, ended up taking any where from two to three weeks dramatically slowing down portlet production. Other problems with actual portlet implementation also slowed us down; where in one particular case several days was needed to figure out a workable solution. Due to all this the final portlet idea, an extension of the Reading List portlet linking it to Amazon.co.uk, was dropped as we thought the time might be better spent fixing and finishing some of the other portlets.

Deployment Script

```
javac $1.java
mkdir portlets/$1/
mkdir portlets/$1/WEB-INF
cp portlets/default/WEB-INF/portlet.xml portlets/$1/WEB-INF/portlet.xml
cp portlets/default/WEB-INF/web.xml portlets/$1/WEB-INF/web.xml
mkdir portlets/$1/WEB-INF/classes/
mv $1.class portlets/$1/WEB-INF/classes/$1.class
sed -i s/classname/$1/g portlets/$1/WEB-INF/portlet.xml
sed -i s/classname/$1/g portlets/$1/WEB-INF/web.xml
sed -i s/keywordses/$2/g portlets/$1/WEB-INF/portlet.xml
jar cvf portlets/$1/$1.war portlets/$1/WEB-INF/
cd ../uPortal_rel-2-6-0-RC2-quick-start/uPortal_rel-2-6-0-RC2/
ant deployPortletApp -DportletApp=/proj/co600/project/javaplets/src071/portlets/$1/$1.war
```

The above code takes 1 argument - the name of the java file you wish to compile.
It then compiles the code and creates a directory to put its class file in.
Then it copies across some default XML files that it searches for a specific string and replaces with the appropriate class name. If a second argument is given, it adds it as a keyword.
Then it packs the entire directory into a War file, then deploys it for use in uPortal.

After we used Netbeans, we found that it compiles by itself compresses it to a .WAR file automatically. Therefore we only needed the following code to deploy the portlet after copying the .WAR file into the correct location.

```
cd ../uportal-2-4-2-myvt-quick-start/uportal-2-4-2-myvt-quick-start/uPortal_rel-2-4-2
ant deployPortletApp -DportletApp=/proj/co600/project/javaplets/Deployer/$1
```

Portlet UML Diagrams

Weather Portlet

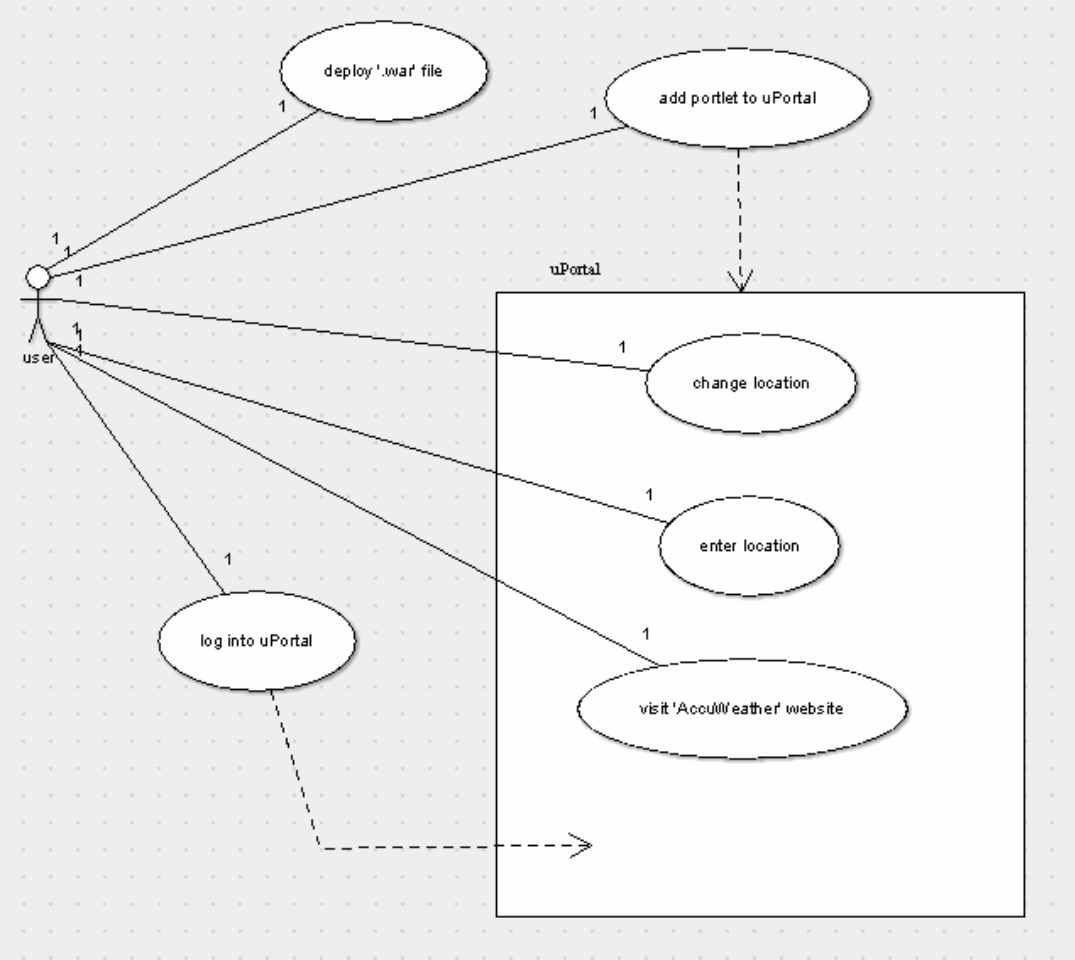


Figure 3: Weather Portlet UML diagram

The user will firstly need to log onto uPortal and add the Weather Portlet. From here they can enter the location that they wish to view, or change the location if they wish to view the weather of another city. As the RSS feed was supplied by AccuWeather.com, there is a link on the portlet that the user can click which will take them to the AccuWeather.com website.

Forum Portlet