Monster Mash – Team Awesome-er
Project Plan

Author:	Joshua Bird, Phil Wilkinson, Thomas Hull, David Haenze, Christian Morgan, Kamarus Alimin, Szymon Stec, Lewis Waldron
Config Ref:	SE_02_PP_03
Date:	02-11-2012
Version:	1.4
Status:	Final

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to describe the project plan for the Monster Mash CS22120 Group Project 2012/13.

1.2 Scope

This project plan aims to set general design goals, high-level software decisions and goals to work towards, given the client’s requirements. All project members are required to read this document to gain an understanding of the project parameters and goal solutions.

1.3 Objectives

The objectives of this document are:

- To indicate the general overview upon which the development of the project is based upon. This should include the choice of platform that the project will be developed on, technologies under-pinning this and the project’s target users;
- to explain how users can interact with the system via a use-case diagram;
- to give a proposed GUI design of the completed project through interface sketches and drawings;
- to give a general time-line spanning the course of the project and proposed start/end dates for different aspects, via a Gantt chart;
- to analyse potential risks that may appear during the development of the project and propose solutions that may help mitigate such risks, by drawing up a risk table.

2. PROJECT PLAN

2.1 Overview

We have been asked to design a “Monster Mash” game in which users can fight, trade and breed their monsters over a network with other users. The users will need to create an account first before they can start playing, and in order for them to return and carry on playing will need to enter their username and password to authenticate themselves as the users. There will also be an element of monetising for the user's to be awarded after winning a fight, and in turn the user can pay for breading partners or new monsters. The monsters can also die in fights and of natural causes. In the game the user will have the ability to search for friends on the game and add them to a friends list, to enable the user to pick fights, or pay for breeding of one of their monsters. The user will only be able to interact with friends on their friends list, their friends will be identified by their email address and be added by a request confirm mechanism.

The game is supposed to be designed as an educational tool to popularise and inform people of evolution and genetics. As this game is supposed to be able be played across a network, we as a team decided on using GlassFish servlets in the developing the game, on Java JDK 1.7. This will provide storage and a communication interface. Additionally, we used GlassFish because the others were very limited in what you could do; e.g. the Google App engine, there were far less packages to use, therefore it prevented us from doing certain things.

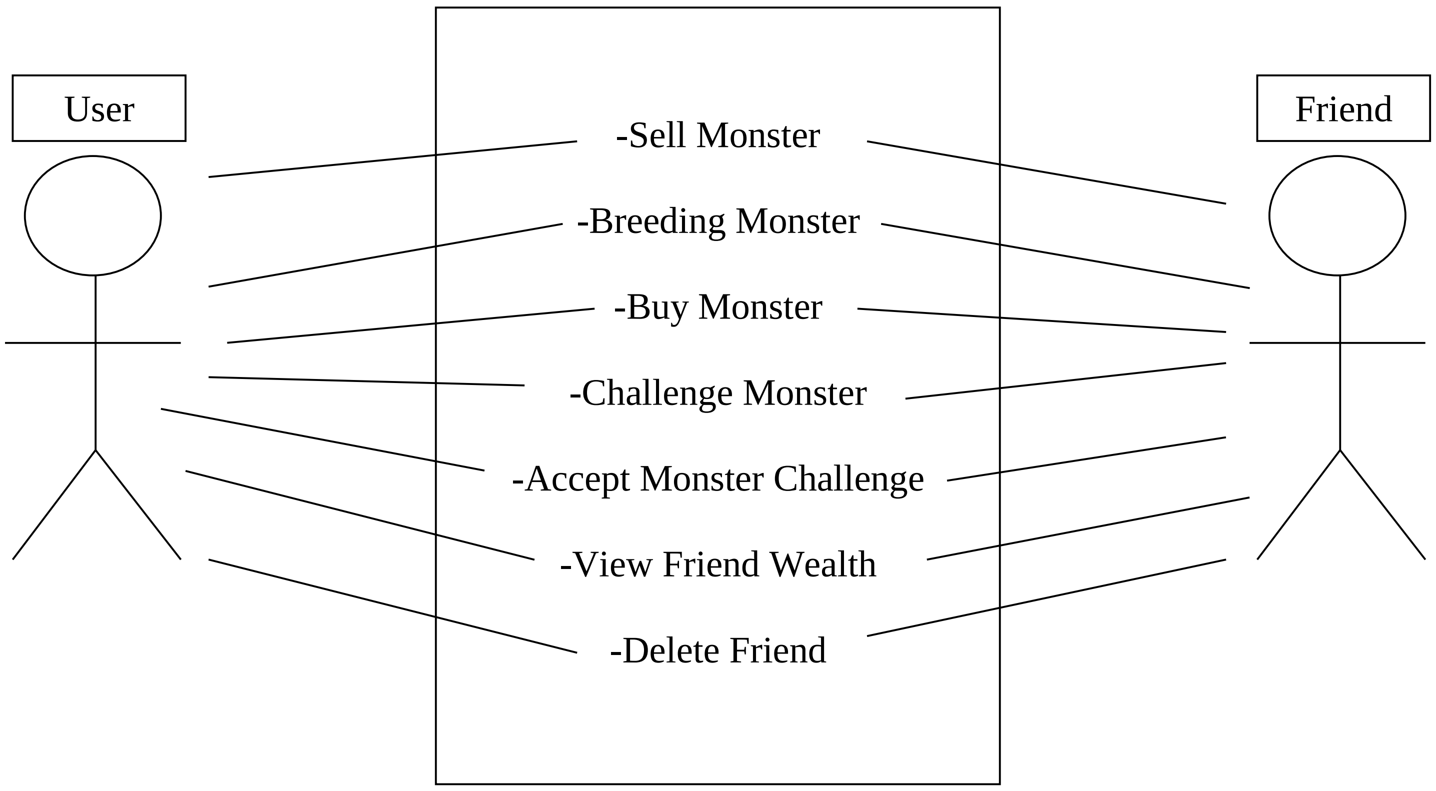
We also have to be able to connect to other users and be able to interact with them and their game. We will standardise the protocols to allow other games on different clients to interact and communicate efficiently. The server will be using the Java Database to store all of the user’s information, as user’s name, their monster’s characteristics and attributes, and who they are friends with in the game. The Java Database will be able to manage all the registered users, allowing new users to be added, and details amended.

We decided on using the Java Database because it is integrated within Java and XML, unlike MySQL which we thought might be more difficult to implement. As this is a web-based user interface requirement we will be using XHTML along with various Java scripts to aid it. The demo-graph we are aiming for with our users will be young people from the age of 14-23 who play web-based games who are interested in monsters or evolutions.

We shall be using a high-level architecture, in which the web page will send and request information via HTTP from the GlassFish servlet. From here the servlet will then communicate with the Java class packages. These packages will then save and load from the Java database which will be using a XML Template, and shall store itself as a XML files. From our discussions we have decided to use GitHub as a source code repository, in order to make sure that we have a clear view on who has updated parts of the project. GitHub supports ticketing which will be useful. With this feature the group leader is able to give out assignments, flag important tasks that need to be completed and can set deadlines. GitHub also comes with managing features, which can limit some team members to only be able to read certain files, and allow others to write to these files - this can be useful in limiting accidents. GitHub also comes with the ability to create Wiki's for the project.

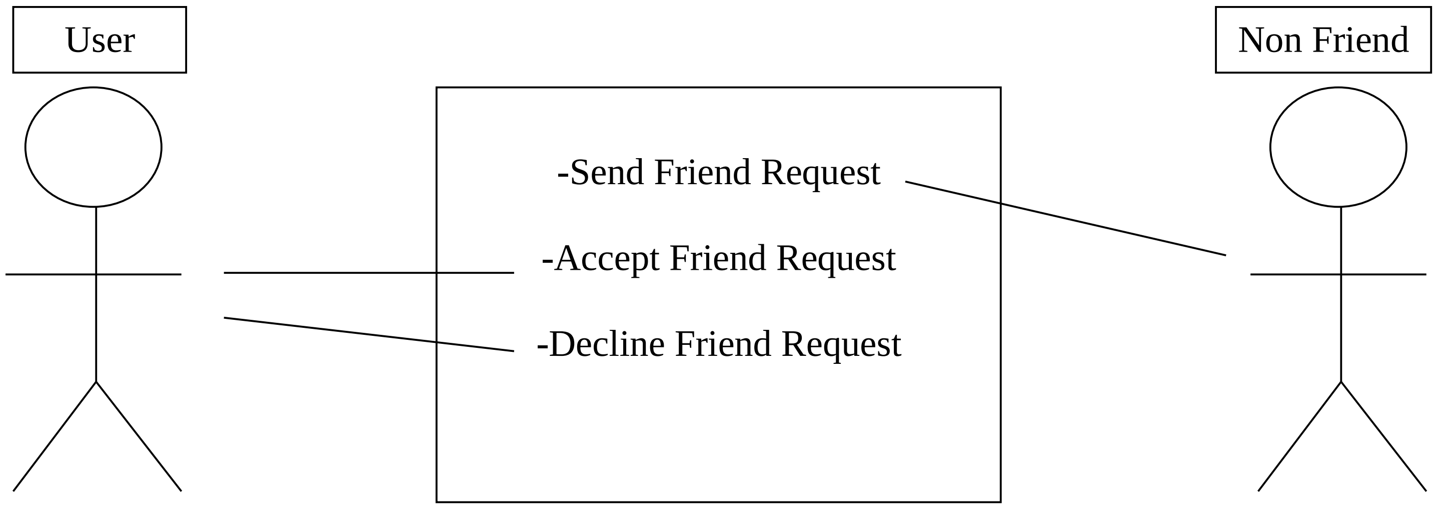
2.2 Use Case Diagram

2.2.1 Use case friend diagram



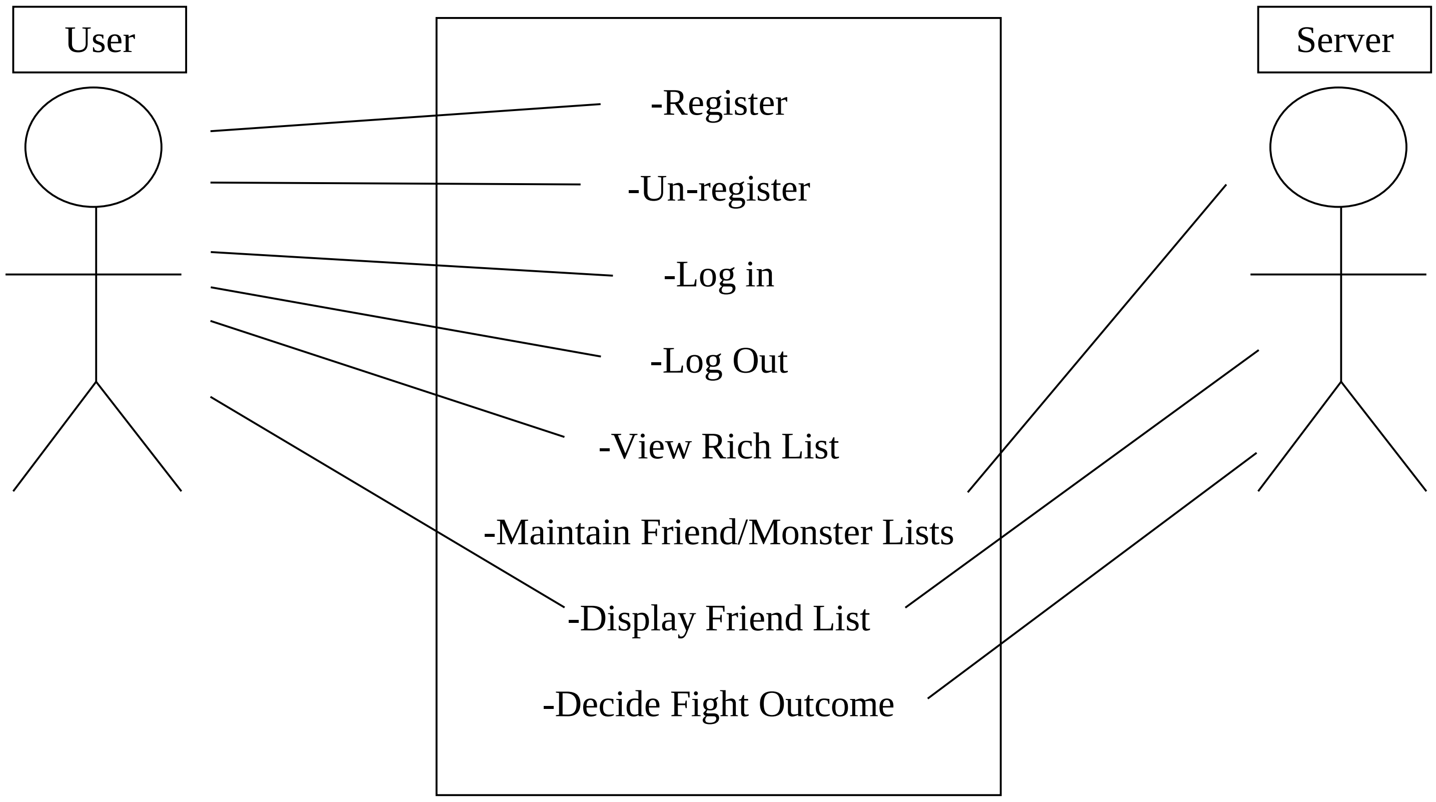
The above use case diagram shows the interactions that users and friend. The both users can sell their monsters to other users. The can also breed monsters with friends monsters, and produce offspring. The users monster can fight other users monsters by challenging each other, this is done by a send request system, were the other user has to accept the challenge. Another feature you can view friends wealth, and users can delete friends.

2.2.2 Use Case Non Friend Diagram



The above use case diagram shows the interaction taken place when users send friend request to each other, and what options the user can take. The user can either accept or decline friend request.

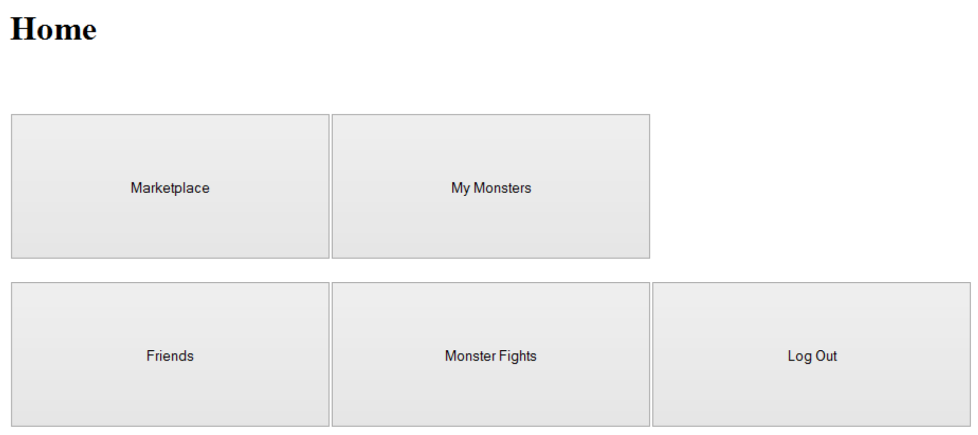
2.2.3 Use Case Server Diagram



The above use case diagram shows the interaction between the user and server. The user can register to play the game. The can also un-register if they wish to, they can also log in and log out from the game. The user can also view the rich list, which shows the wealthiest players in the game. The server also maintains the friend and monsters list in the game. The user can also display friend list while the sever can also decide fight outcome between two monsters.

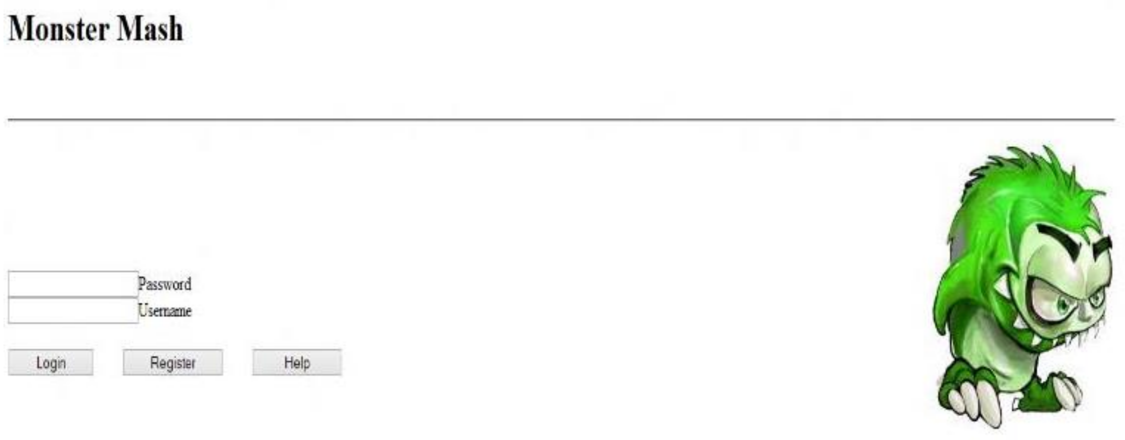
2.3 User interface design

2.3.1 Home screen



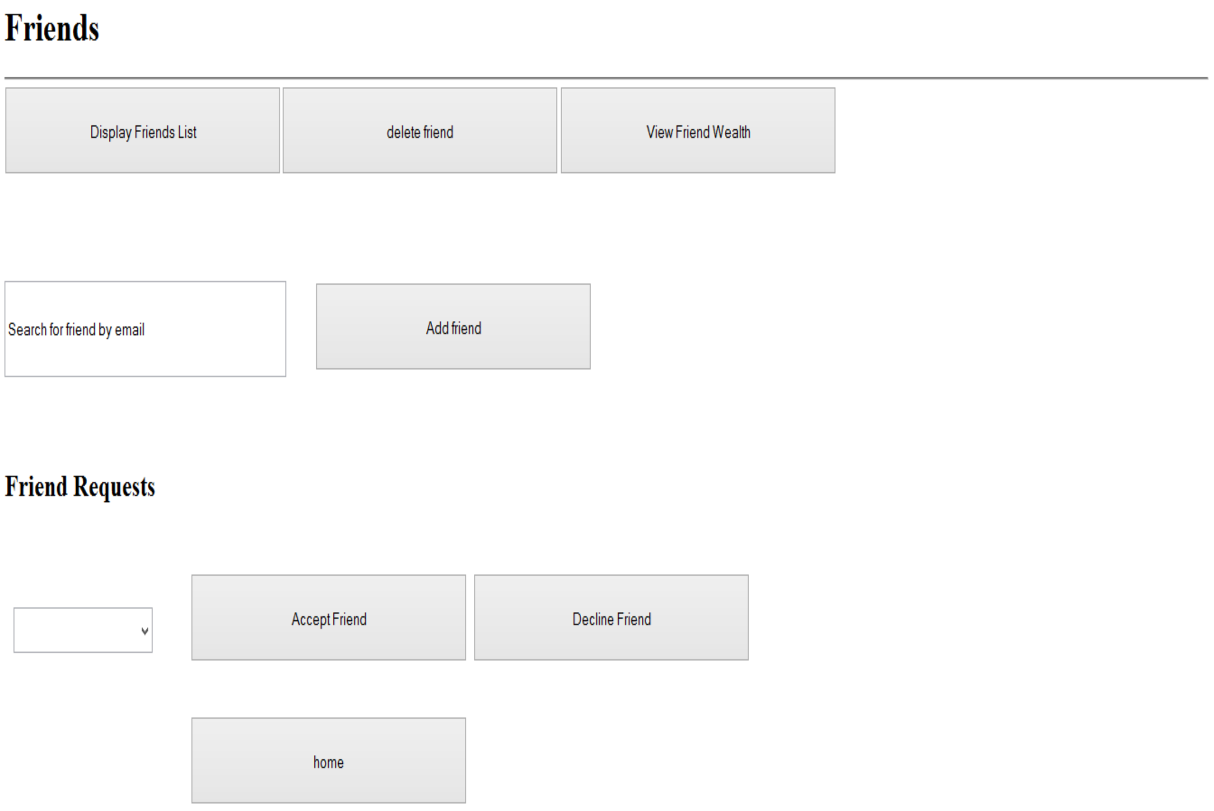
After logging in with the correct credentials, the next page to the user will be the home page.
Each button will take the user to the following:
Marketplace > marketplace.html (where monsters are bought and rented)
My Monsters > myMonsters.html (where one can breed, sell and loan monsters)
Friends > friends.html (used to see and delete current friends as well as add new ones and manage friend requests.
Monster Fights > monsterFights.html (To challenge monsters of other friends)
Log Out > login.html (Takes the user back to the log in page.)

2.3.2 Login screen



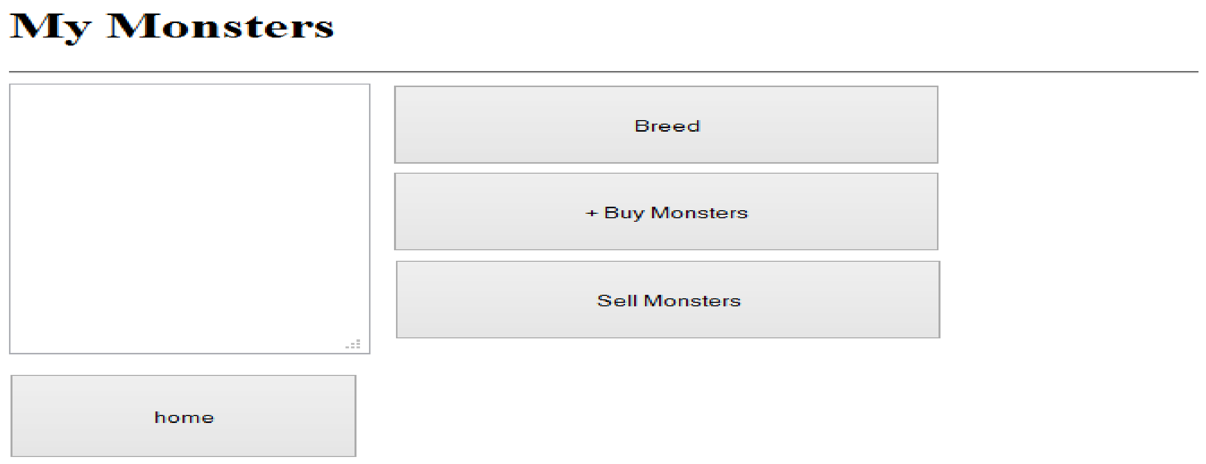
When the game is started the first page the user should see is the log-in page. As the user can see here, one can log-in with their correct credentials. If the user were not to have any, they can register (or view the help page).

2.3.3 Friends



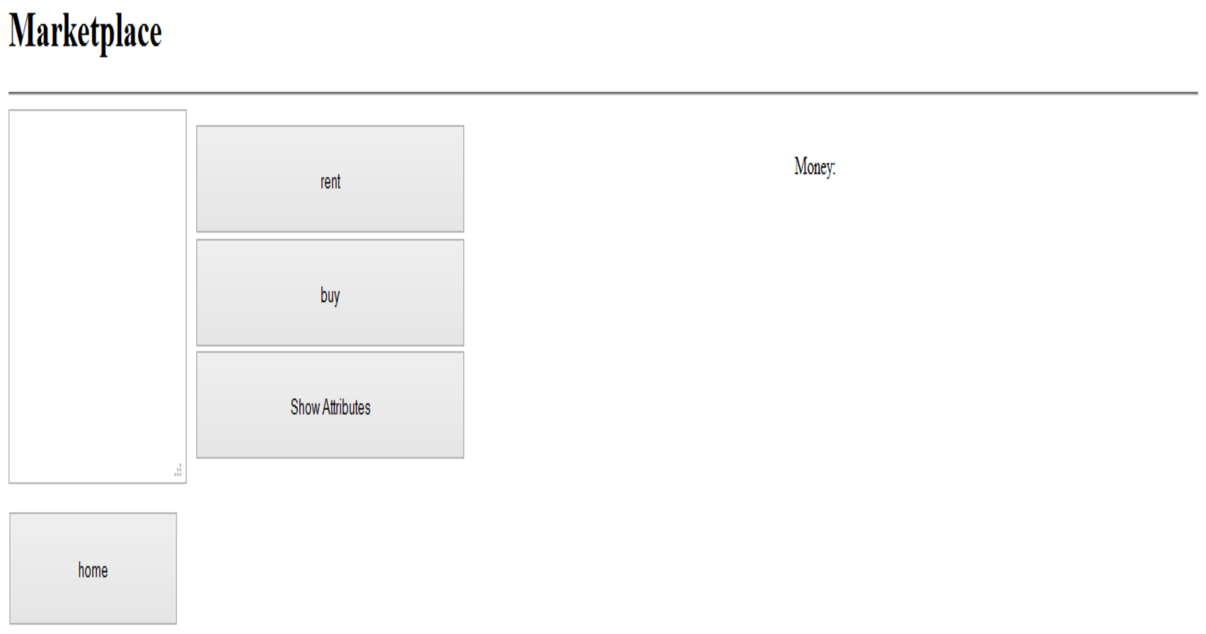
After clicking the ‘friends’ page the following page will be displayed. In this page one can manage their current friends by deleting and adding them. The ‘display friends list’ will bring up a list of all currently connected friends. ‘View friend wealth’ will show a rich list of the richest players in terms of money in the game. By entering a valid user email address users will be added via the ‘add friend’ button. Friend requests will show in the drop down box towards the bottom left of the screen shot. By selecting the appropriate request it is easy to see how the user will accept or delete the friend request. As per usual the home page takes the user back to the home page.

2.3.4 List of user’s current monsters



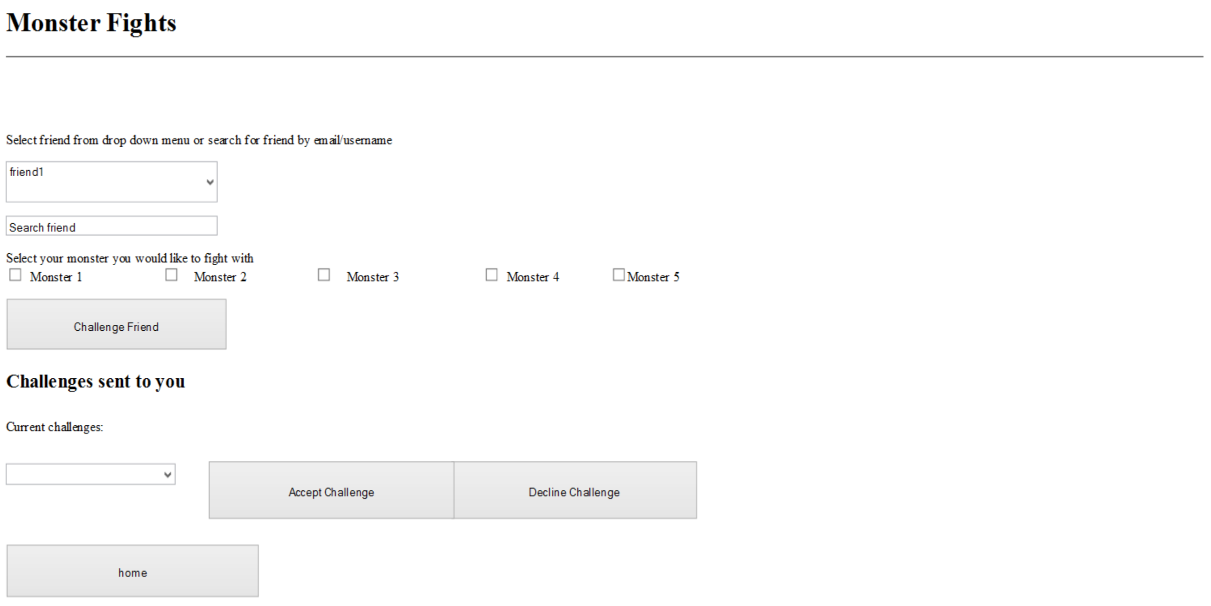
After clicking the ‘my monsters’ button on the home page the following page will be shown. A user will be able to breed a monster, as when the button is pressed, a prompt box with a drop-down box of your monsters will be shown, enabling the user to choose which monsters to breed. The ‘buy monsters’ button will take the user back to the marketplace page. As for the ‘sell monsters’ button, a prompt box will display, enabling the user to specify whether they want to sell a monster or loan it out to another user, and for how much money. The home button takes the user back to the home page.

2.3.5 Buy a new monster



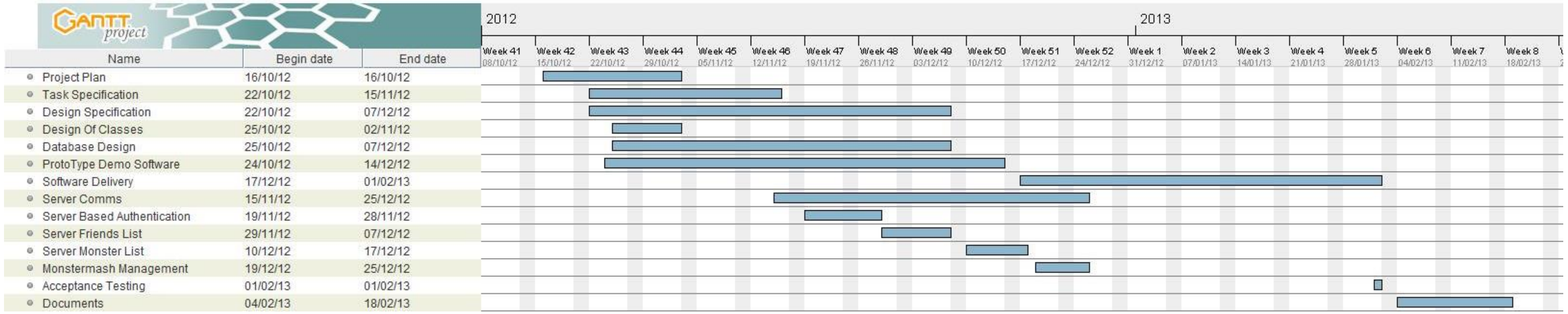
After clicking the marketplace button on the home page this page shall load. There is a list box on the left-hand side which will display monsters that other friends are selling or loaning out. After clicking on the monster, the user shall be able to rent or buy them by clicking the appropriate buttons. The ‘show attributes’ button will tell the user more general info about the monster. At the top right corner of the screen where it says “Money:”, this will show the funds available to the user. The home button will take the user back to the home page.

2.3.6 Fight a monster



Via the 'monster fights' button on the home page this page shall be reached. To challenge a friend the user can either search for them by their email, or select them from the drop down menu. The user will then select which monster to fight with and click 'challenge friend' - a pop up box will then display the fight outcome. Challenges sent to the user can also be managed in a very similar way to friend requests. Current challenges are displayed in the drop-down menu, and with the appropriate buttons, one can accept or decline the challenges.

2.4 Gantt chart



Name	Begin date	End date
Project Plan	16/10/12	16/10/12
Task Specification	22/10/12	15/11/12
Design Specification	22/10/12	07/12/12
Design Of Classes	25/10/12	02/11/12
Database Design	25/10/12	07/12/12
ProtoType Demo Software	24/10/12	14/12/12
Software Delivery	17/12/12	01/02/13
Server Comms	15/11/12	25/12/12
Server Based Authentication	19/11/12	28/11/12
Server Friends List	29/11/12	07/12/12
Server Monster List	10/12/12	17/12/12
Monstermash Management	19/12/12	25/12/12
Acceptance Testing	01/02/13	01/02/13
Documents	04/02/13	18/02/13

2.5 Risk Assessment

Risk assessment is the act of determining the probability that a risk will occur and the impact that event would have, should it occur. This is basically a “cause and effect” analysis. The “cause” is the event that might occur; while the “effect” is the potential impact to a project should the event occur.

Risk No.	Threat	Risk Description	Mitigate Actions
1	Aberystwyth’s servers crash	Machine crashes, losing the project work.	Mitigated by backing up after work with a pen drive or other storage device. While also keeping records on ‘GitHub’
2	Health risk	Lead coder or one of the members falls ill.	Mitigated by setting up regular meetings between other coders to explain the current progress. Also well documented code so other coders can understand what is going on with the project.
3	Project deadline	Work not completed on dates set.	Have target deadline set on the week before “official” deadlines.
4	Copyright	Copyright infringement or violation of the copyright law. For example, the use of an image which has watermarking embedded download from the internet.	Mitigated by obtaining the author’s permission or give appropriate credit to the author. Or by finding license free images.
5	Sabotage by another group	Project might be sabotaged by another group due either pranks or malicious attack	Mitigated by seeking the higher authority for the action to be done
6	Communication barrier	Miscommunication among member might slow down the process of the project work.	Use Appropriate channel of communication or medium to convey the message better.
7	Act of nature	Unforeseeable event that is outside our control, no one held responsible for what have been done.ie Natural event such as hurricanes, apocalypse and etc.	Unfortunately, no mitigation possible.
8	Rescheduling Informal meeting	If a member unable to attend to an informal meeting due to lectures, social activities and sport activities.	The project leader should coordinate a common free time that everyone will agree to attend.
9	Some or all group members are not used to group projects	Some people are unaccustomed to working in a group and feel uncomfortable participating or initiating group discussion.	Mitigate by assigning roles to each member in which they feel responsible for the project. Besides that, an ice breaking activity such as going to a pub etc. on the weekend is a good way of getting to know each member.
10	If a group member withdraws from the course	This could lead to chaos and management problem within the project group. Illness, accident or family obligations are some of the possible reasons for the withdrawal.	Seek the lecturer assistance to adjust scope of the project. Project leader should reorganize roles and task to compensate for the loss of manpower.
11	Wet Floor	If the floor has been cleaned in the university campus and a member trips and causes an injury.	Mitigate by being observant for wet floor signs while around the university campus and also as mentioned previously have other members with knowledge of the project to replace the member if needed.
13	Work being copied	If a member is in a public place working on the project someone could watch them and copy work.	Members should be observant while working on the project in public places and try and make sure sensitive data isn't lost to others.
14	User not familiar with tools	If a member does not know how to use GitHub or an IDE we are using to work on and is unable to do the work because of it, this could lead to a loss of work done.	Any group member unsure how to use any of the software should ask for help from others.
15	More than one user trying to update a file at the same time	If more than one user edits a file on GitHub and tries to save, someone else's work could be overwritten.	Making use of GitHub's ticketing feature to be able to keep track of updates to files.

16	Stress	Any group member could have their work affected by stress, be it work related or otherwise.	Those being effected should notify the project manager or seek help from the university in order to deal with it. The project leader can also be spoken to if the member feels like they have too much work to do, so that it can be reviewed.

REFERENCES

[1] Software Engineering Group Projects: General Documentation Standards. C. J. Price, N. W. Hardy. SE.QA.03. 1.5 Release

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	20-10-2012	Created Initial Document	PW
1.1	N/A	25-10-2012	Added draft versions of sections to the body	JB
1.2	N/A	02-11-2012	Updated to latest versions of sections.	JB
1.3	N/A	28-01-2013	Updated use-case	CM
1.4	N/A	28-01-2013	Updated gantt chart	PW