Software Engineering Group Project
Project Plan

Authors: Group 12
Config Ref: SE_G12_PP_00
Date: 30-10-2012
Version: 1.0
Status: Release

CONTENTS

1. INTRODUCTION

This is the project plan for the group project.

1.1 Purpose of this Document

The purpose of this document is to create and accurate translation of the clients requirements into to a set of objectives and deliverables. This document includes designs of the final programme and descriptions of how it will interact with a user. This document also lays out the way in which the project will be completed

1.2 Scope

This document aims to provide a plan as to how we will move forward and complete the project, using the Gantt Chart and Risk Analysis Table to plan the timing and distribution of layout of the project, and the Use-Case diagrams and Interface designs to specify how the finished project will work.

1.3 Objectives

The objective of this document is to create a high level description of our project.

2. OVERVIEW

2.1 Intended Users

Monster Mash is a game primarily aimed at young people within the general populous. The game is designed to not only be fun, but to be used as an educational tool to help the players understand evolution and genetics. As a web-based game, Monster Mash is designed to tap into the growing reservoir of young internet users.

2.2 Intended Technologies

2.2.1 Java

Java is a high level object orientated programming language, this is the language that the project will be implemented in as per the requirements specification .[1]

2.2.2 GlassFish

Deployment

GlassFish and Tomcat are both written in Java, and the servlets if GlassFish are based on a earlier version of Tomcat, so all web services that work on Tomcat should also be able to tun run on GlassFish

Administration and Maintenance

GlassFish includes a web interface for maintaining the server, including things like databases, http load balancers and other tools to help with maintenance and scaling.

GlassFish is owned by Oracle and is maintained by both Oracle and the development community, so it should stay well updated with current technologies for the future. Tomcat on the other hand is only maintained by the community, and should community move on the rate of updates and bug fixes might be slow, compared to GlassFish which has Oracle working on it.

Performance

There is not much difference in performance between Tomcat and GlassFish.

Development

Many of the other groups has decided on using GlassFish, so there might be a advantage for us to also go for GlassFish as it might make the standardisation with other groups easier. GlassFish should also be easier to work with when using University facilities compared to Tomcat where a set-up might be required. If the application is built without the use of any of the JavaEE features provided by GlassFish reverting to Tomcat at a later point should not be impossible without throwing away the whole code base.

2.2.3 Git

What is Git?

Git is a version control system that is distributed. It does not depend directly on having a connection to a central server, as working directories contain the repositories. This means revisions and history are fully accessible without access to a central server.

For a central server we will be using a GitHub private repository, as the university doesn't have Git set up for group usage.

Why we're using it instead of SVN

The biggest reason is that at least 2 members of our group are already knowledgeable on how to use Git – this both saves the two people learning a totally new system, when it's easier for those with the knowledge already to show the rest of the group both how to use it and be able to maintain it reasonably well.

Another reason is some mistakes in Git are less fatal than they are in SVN. If the repository on an SVN server is deleted, then everything is gone(unless regular backups are made). In a Git system, as everyone has a full repository on their working machines it is very hard to delete all the work accidentally. So if someone somehow deletes the repository on both GitHub and their own machine, someone else with the most recent version can push it back to GitHub, and it can be pulled back down by the person who accidentally deleted their own copy.

3. USE CASE

3.1 Diagram
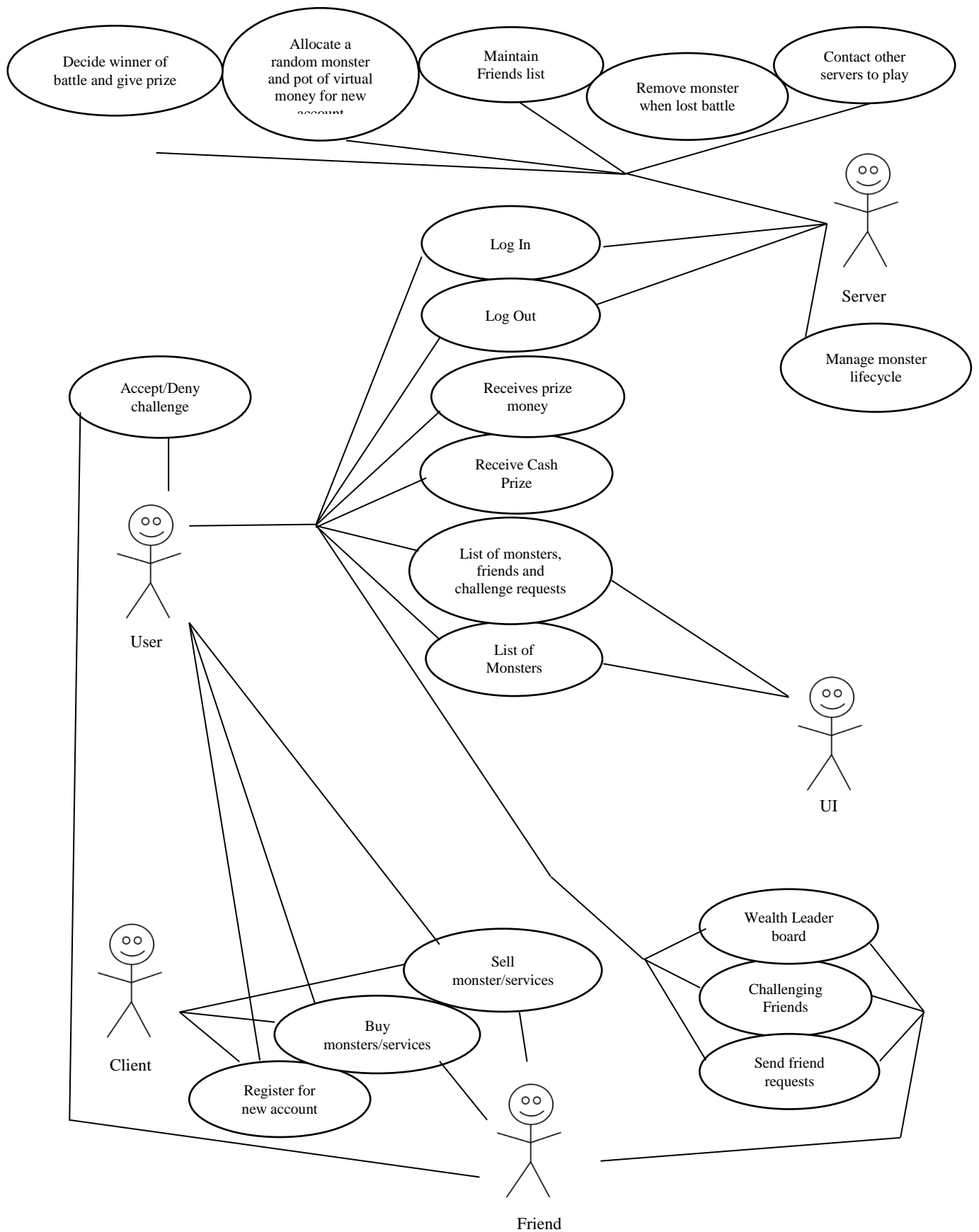
3.2 Interactions

There are many interactions happening in the previous UML Use Case diagram, using the actors Server, User, UI, Client and Friend.

All of the friend links to nodes depend on the user and vice versa, for example if the user sent his/her friend a friend request, it then requires the friend to accept or deny zed request. Similar with challenging for a fight, or even breeding monsters. If the friend requests to buy the users monster, this requires the user to firstly put up the monster for sale for the friend to browse. User and friend are key nodes in this diagram, since a lot of events can happen between the two.

The client side is a small section, yet an important section for the user and friend. Since without the client, neither user or friend would be able to interact with the functions between the two. For example you wouldn't be able to buy your friends monster, or ask to breed, if it wasn't for the client.

Implementing certain interfaces requires the user, friend and UI to interact with one another. For example, looking at the Wealth Leader Board, this would require the UI to display the leader board, which in turn would be empty without the user data and friend data. This includes having a list of monsters and the friend list. Effectively this could be implemented outside of the UI; however it seems more logical to have it embedded.

Certain server links require a prerequisite before the function can run. It does a lot for the game behind the scenes, contacting other servers for the ability to play is a large issue, if there is no servers, then there is no game. It generates the log in and log out, with the storage of data and allocates a new registered user to the game with a random monster and virtual money, thus allowing them to play the game. A lot of the server functions don't rely on another node to be working with them; they are standalone, just like maintaining the friends list, or removing a dead monster when it has lost a battle.

3.3 Usage Examples

Taking a random node and working out where it will go, and what it will interact is the most useful part of a UML Use Case diagram. As some examples:

Accepting or Denying a challenge – In this example we will assume that the user is sending the request and the friend is choosing to accept or deny the challenge to a fight. The user sends out the battle request, and then waits on the friend to accept or deny. Which similarly if on the use case we start from the Friend node, we go to Challenging friends (challenging the user) which creates a link to the user.

Selling monsters/services – In this example we will assume that the friend is the one selling the service. There are three main links going into this function; Client, Friend and User, each with their own unique links to the function. Firstly you need the client to be able to Sell the monsters or services, which allows the interaction between the player (user) and his/her friend.

Lists of monsters, friends and requests – This is quite a large function which could be the heart of the program. This allows the UI to generate the friends list, which in turn allows the user to interact with to challenge friends. The lists of monsters allow the user to keep track of his/her monsters injuries, attributes and even to be able to put them up for sale. Finally having a challenges list is useful because it works as a notification. There is no reason to have the ability to be challenged by someone if you don't know something is pending, same goes for buying and selling/breeding.
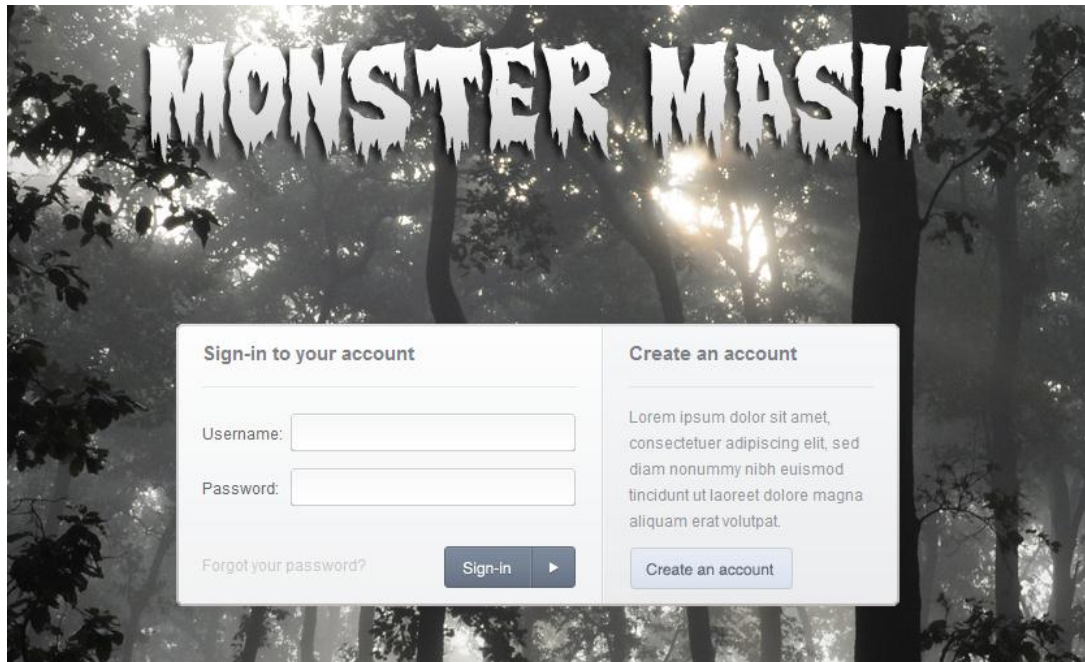
4. USER INTERFACE DESIGNS

4.1 Final Digital Designs

The background of the webpage designs is taken from an open image repository.[2]
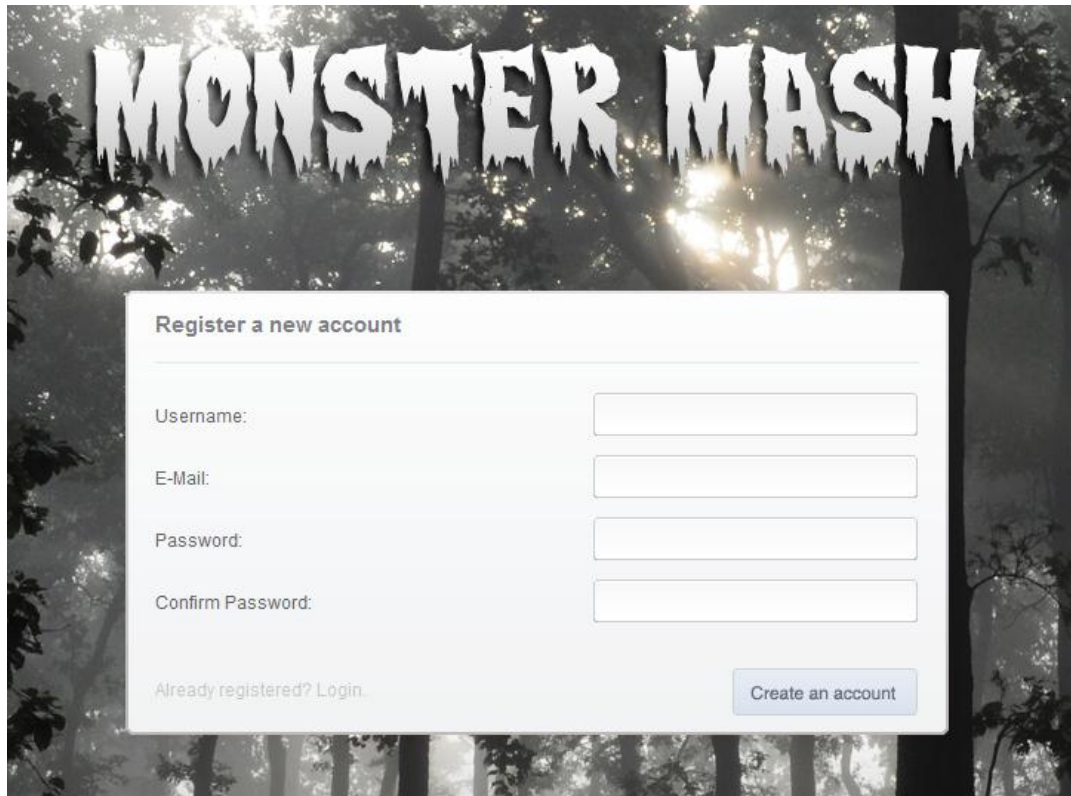
The icon images used in the designs is also taken form a free to use image source. [3]

4.1.1 Login Page

When user opens Monster Mash game, first page which shows up is **Login** page, where user can login onto their account and then it will be redirected to **Main** page (by clicking "*Sign-in*" button). There are also two hyperlinks to **Register** page ("*Create an account*" button) and **Forgotten Password** page ("*Forgot your password?*" link).

4.1.2 Register Page

Register page is a place where user can create new account. After clicking "*Create an account*" button, the form is processed. This is where a user will enter their data when they are usign the sit for the first time

4.1.3 Main Page
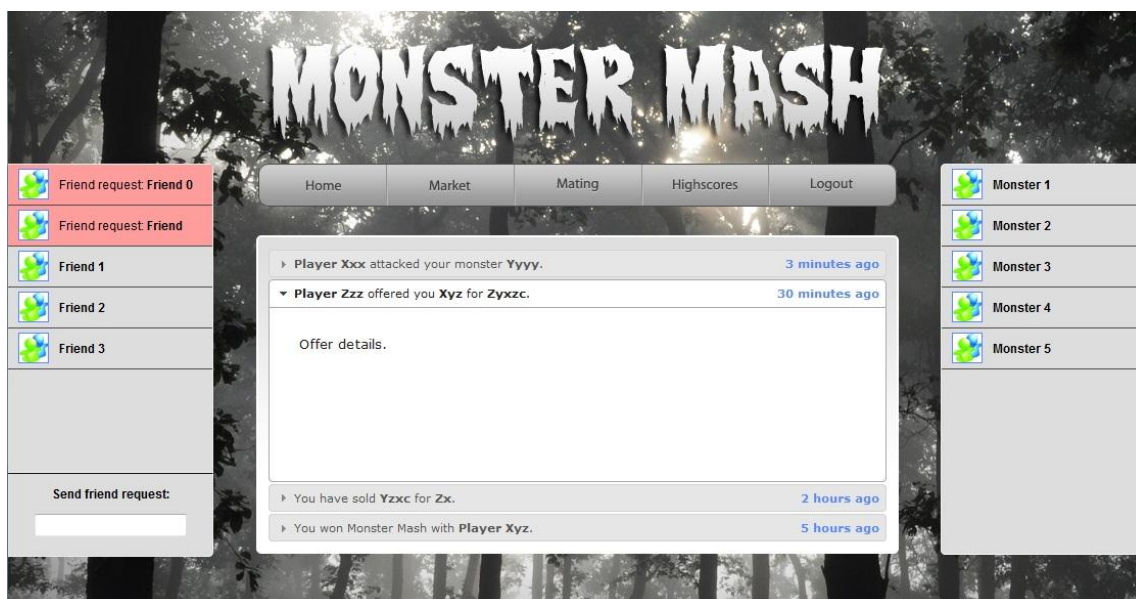
After logging onto their account user sees the **Main** page. **Main** page consists of top menu, 2 sidebars (Friendlist and Monsterlist). The central panel on the main screen displays the updates of reent actions by the user or their friends. On the other pages only the central panel changes, the Friendslist and Monsterlist remain in place

4.1.4 Menu Bar

Each menu element redirects logged user to different subpage. Only "*Logout*" element logs out user and redirect to login page. It is on these subpages where the different central panel is displayed that allows the user to interact with the website.

4.1.5 Friends List
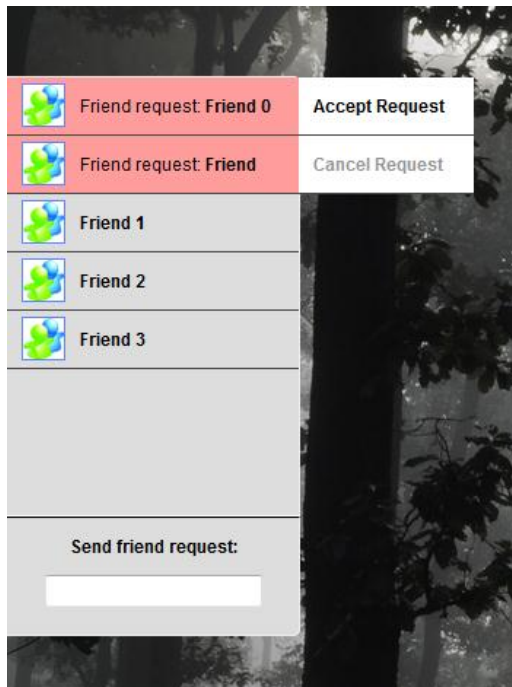
Friendlist is a sidebar, where we can see all friends, all friend requests or even send requests. Clicking on friends in the list allows you to interact with them by challenging them to battles or offering mating. Clicking on pending requests allows a user to either accept or reject the requests. The field at the bottom allows users to search for their friends by typing in their email address so they can send them a friend request.

4.1.6 Monster List
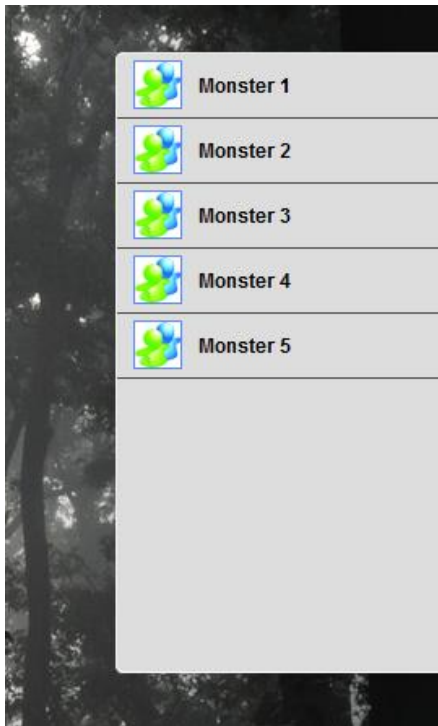
Monsterlist is a second sidebar, where we can see a list of all our monsters. Hovering mouse over single element of the list shows small information about monster's attributes.

4.1.7 Update Panel
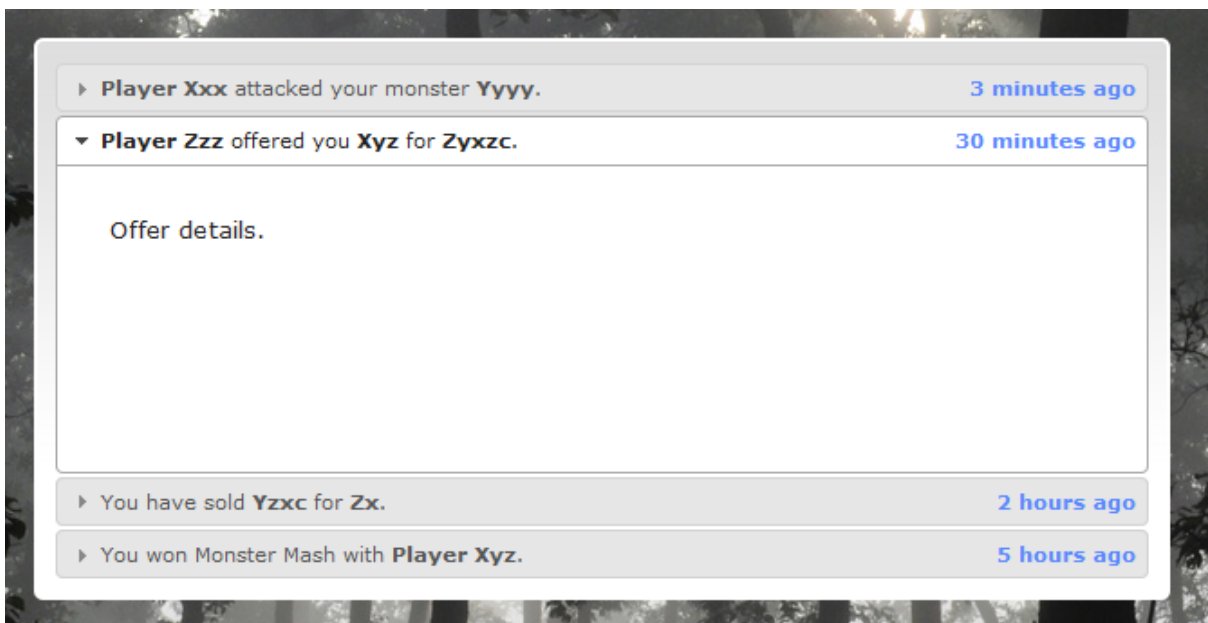
This is the update panel displayed on the main page with information on recent events and current offers.

4.1.8 Market
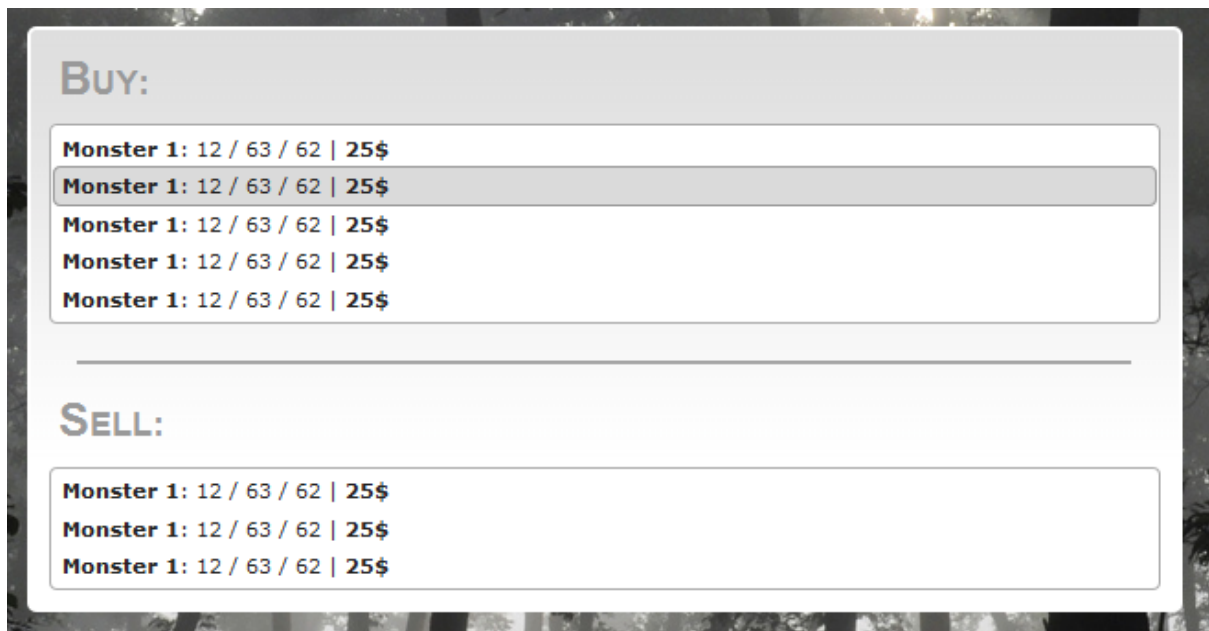
Market is a place where users can buy new monsters or sell their own monsters. Clicking on each offer shows small window with the genetic traits of monsters and other statistics.

4.1.9 Mating

Mating page is similar to the **Market** page, but it allows you to offer your male monsters or find male monsters to mate with your female monsters.

4.1.10 High Scores
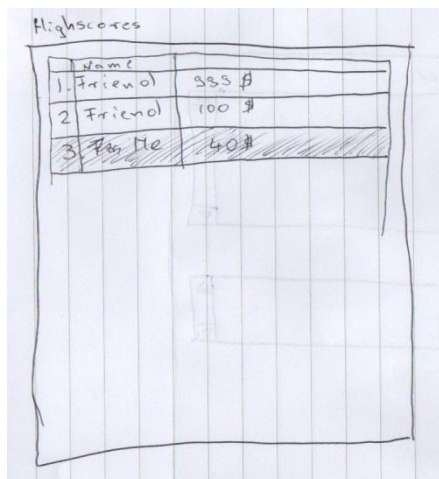
Highscores is a table with the richest players on our server, ordered by wealth

4.2 Initial Sketch Designs

These are the original sketches that were drawn before the digital versions were made, these demonstrate our design process.

4.2.1 High Scores

4.2.2   Login Screen

4.2.3   Main Screen

5. GANTT CHART

Second Semester

First Semester

First Semester — Gantt Chart

	Begin Date	End Date	Duration
Interaction Design	18/10/2012	29/10/2012	11
Introduction	18/10/2012	29/10/2012	11
Overview	18/10/2012	29/10/2012	11
Use Case Diagram	18/10/2012	29/10/2012	11
User Interface Design	18/10/2012	29/10/2012	11
Gantt Chart	18/10/2012	29/10/2012	11
Risk Analysis	18/10/2012	29/10/2012	11
Test Specification	29/10/2012	13/11/2012	15
Desgin Specification	29/10/2012	03/12/2012	35
First ProtoType	05/10/2012	10/12/2012	66
Further Implementation	10/12/2012	21/01/2013	42
Integration and Testing Week	28/01/2013	01/02/2013	4
Delivery Of Software	01/02/2013	01/02/2013	0
Acceptance Testing	01/02/2013	01/02/2013	0
Document Hand In	04/02/2013	18/02/2013	14

| W1 15/10/2012 | W2 22/10/2012 | W3 29/10/2012 | W4 05/11/2012 | W5 12/11/2012 | W6 19/11/2012 | W7 26/11/2012 | W8 03/12/2012 | W9 10/12/2012 | W10 17/12/2012 | W11 24/12/2012 |

Second Semester — Gantt Chart

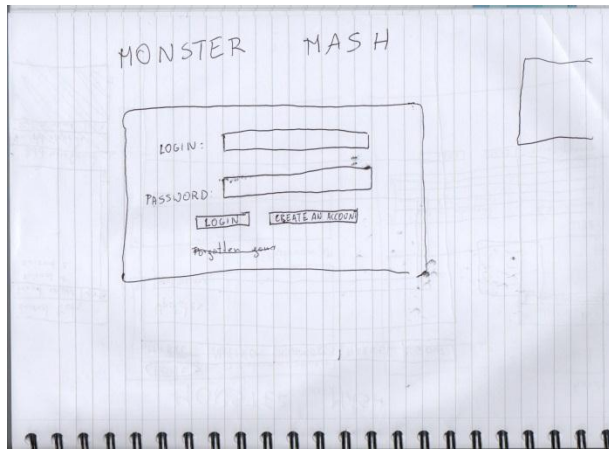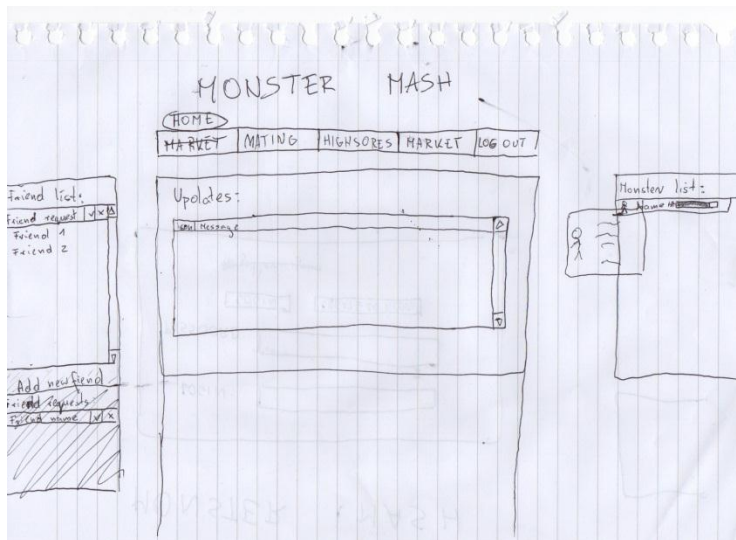|	Begin Date	End Date	Duration
Interaction Design	18/10/2012	29/10/2012	11
Introduction	18/10/2012	29/10/2012	11
Overview	18/10/2012	29/10/2012	11
Use Case Diagram	18/10/2012	29/10/2012	11
User Interface Design	18/10/2012	29/10/2012	11
Gantt Chart	18/10/2012	29/10/2012	11
Risk Analysis	18/10/2012	29/10/2012	11
Test Specification	29/10/2012	13/11/2012	15
Desgin Specification	29/10/2012	03/12/2012	35
First ProtoType	05/10/2012	10/12/2012	66
Further Implementation	10/12/2012	21/01/2013	42
Integration and Testing Week	28/01/2013	01/02/2013	4
Delivery Of Software	01/02/2013	01/02/2013	0
Acceptance Testing	01/02/2013	01/02/2013	0
Document Hand In	04/02/2013	18/02/2013	14

| W13 07/01/2013 | W14 14/01/2013 | W15 21/01/2013 | W16 28/01/2013 | W17 04/02/2013 | W18 11/02/2013 | W19 18/02/2013 | W20 25/02/2013 |

6. RISK ANALYSIS

Risk	Possibility	Solution
Illness	high	We have deputies (more than one person is responsible for each thing)
All accidents involving permanent or long absence	extra low	Same as illness
Not turning up to a meeting	extra high	Same as illness + emailing minutes
Decided to do nothing	low	Same as illness
Intellectual property right problems	very low	Using only free-for-use property + having links to every free property used
More rework than anticipated	high	Doing a little more from the start(to prevent from happening) + throwing more people at such task later
Somebody steals our idea	low	Reporting, showing project history
Technical problems on a presentation day	high	Having extra equipment ready + using other
Loss of data	high	Recovery from a repository
Sudden change of specification	low	Taking actions to apply these changes
Misunderstanding of specification	mid	Multiple persons should read it + re-read it if needed
Not working or incorrectly working buttons/links	high	Testing every button and link before "release" + applying fixes
Incorrect attribute calculation (health, strength...)	high	Unit testing
Disallowed characters in names	high	
Server to server communication problems	high	Using correct ones from the start + limiting user input to those that are allowed
Client to server communication problems	high	Checking the connection and compatibility
Time difference / synchronization problems	Very low	Using only server's time

REFERENCES

[1]   **Software Engineering Group Projects Monster Mash Game Requirements Specification**
 Config Ref: SE.CS.RS

[2] **Stock.XCHNG**   http://sxc.hu/

[3] **"Bright" Icon set by Min Tran**   http://min.frexy.com/

DOCUMENT HISTORY

Version	CCF No.	Date	Changes made to document	Changed by
0.1		29/10/2012	Document first created addition of most major elements.	James Upshall (jau1)
0.2		31/10/2012	Overview elements added, uses-case updated, design images updated and descriptions added, Risk Analysis Chart updated.	James Upshall (jau1)
1.0		01/11/2012	Inclusion of Gantt Chart and update to release version	James Upshall (jau1)