**CS22120 Software Development Life Cycle**

**Group 05 Project Plan**

**Author:** All

**Config Ref:** SE\_05\_Interaction

**Date:** Friday 26th October 2012

**Version:** 1.3

**Status:** Draft

Table of Contents

1. Introduction 3

1.1. Purpose of this Document 3

1.1.1. Scope 3

1.2. Objectives 3

2. Overview of Proposed System 3

2.1. Technology Being Used 3

2.2. Use-cases 4

2.2.1. Use-case for Users 4

2.2.2. Use-case for Server 5

2.3. User Interface Design 5

2.3.1. User Interface Layout Designs 6

2.3.1.1. Sign Up 6

2.3.1.2. Log In 7

2.3.1.3. Menu Bar 7

2.3.1.4. Cash Pile 8

2.3.1.5. Banner 8

2.3.1.6. Homepage 8

2.3.1.7. Friend's Stable 9

2.3.1.8. Add Friend 10

2.3.1.9. Shop 10

2.3.1.10. Battle Report 11

2.5. Risk Assessment Table 12

2.6. Task Allocation 15

2.7. Gantt Chart 15

3. References 15

4. Document Change History 15

# 1. Introduction

## 1.1. Purpose of this Document

The purpose of this document is to describe how we plan on creating the web application from the design to the technology we plan to use.

### 1.1.1. Scope

This document specifies the user interface designs and what technology will be used.

## 1.2. Objectives

The main objective of this project plan is to show to the customer how we plan on creating the system. Showing the design of the UI will give the customer an idea of how Monster Mash will look. Describing what technology that will be used will give the customer an insight of the “back end” side of the application.

# 2. Overview of Proposed System

## 2.1. Technology Being Used

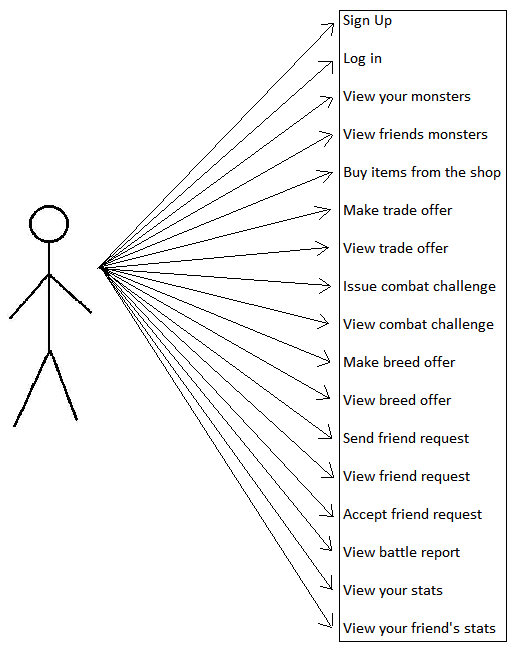
At first we were thinking of using Tomcat server application, because it supports JSP and Java Servlets and is lighter than every other server application. We decided to use Glassfish, as it is being used by all the other groups and is the only software supported by the university. Glassfish provides full Java EE support including JSP and Java Servlets, which we will be using and is easier to navigate through than alternative software.

Glassfish is an open – source application server by Sun Microsystems. It provides full support for Java EE, JavaBeans, JPA, JSF etc. Glassfish has many more administration and monitoring tools than alternative software, such as Tomcat. We will be using Java Servlets to deal with requests from the client on our server. Considering we are striving for an MVC design pattern, Servlets are the right choice for control i.e handling requests.

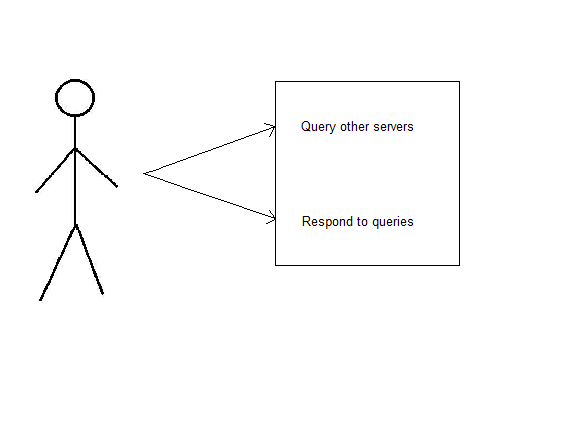
A Java Servlet is a Java class used to extend the capabilities of the server. Although Servlets can respond to any types of requests, they are commonly used to extend the applications hosted by web servers, so they can be thought of as Java Applets that run on servers instead of in web browsers.

## 2.2. Use-cases

### 2.2.1. Use-case for Users



### 2.2.2. Use-case for Server



## 2.3. User Interface Design

To begin the user interface design, we started with a use case diagram which had everything the user needed. From this we created a directed graph which shows all the options the user has for navigation from that page. After agreeing on the pages we will be creating, we started with a basic hand drawn design of what the game will look like. Once we had produced the hand drawn designs, it was necessary to put them in to a digital format as well as annotated versions. There will be seven pages, which include:

* Log In
* Sign Up
* Homepage
* Friend's Stable
* Shop
* Add Friend
* Battle Report

### 2.3.1. User Interface Layout Designs

#### 2.3.1.1. Sign Up

This will check

the database to see

if the email which has

been entered, is not

already taken.

These fields must match.

This will send the data to the

database and will be the user's

log in credentials. This will direct

the user to the “log in” page.



#### 2.3.1.2. Log In

### 

If the user enters the

Correct email and password,

they will be taken to the homepage.

If not, an error message will come

up saying incorrect email/password.

The email and password

that the user entered upon sign

up will be their credentials.

Non-registered user's will

click this to sign up. This will

direct them to the sign up page.

#### 2.3.1.3. Menu Bar

Each page (that the user is signed into (not sign up or log in page)) will have a menu bar. The menu bar will appear below the banner and will be on each page they visit. On this bar, the actions are undecided.

It will have some combination of:

* Home button - navigate the user back to the home.
* Cash Pile – shows the amount of virtual money the user has.
* Friend Request - any requests from people to add/accept.
* Breeding Offers - any offers from friends to breed.
* Sales - any offers from friends to buy (buying a user's monster).
* Shop – buying a starter monster.

*Below is an example of the menu bar.*

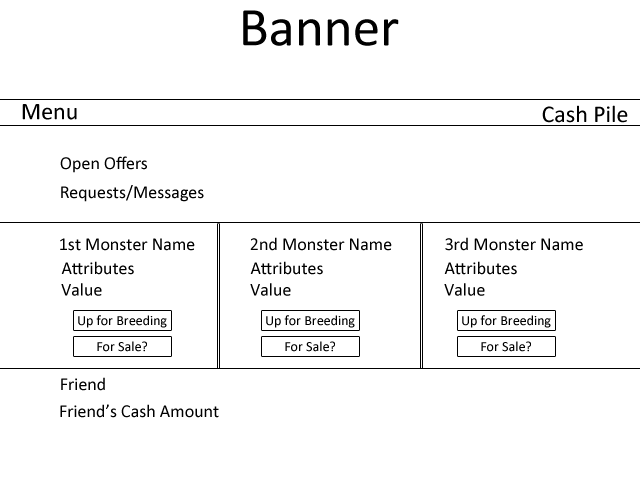
#### 2.3.1.4. Cash Pile

It is one of the requirements for the friends list to be ranked and this can be done by the highest amount of money. Each user will have a database and will be able to be sorted from richest – poorest. This can be added to the homepage once agreed on.

#### 2.3.1.5. Banner

Each page will have a banner which will be a design (text/image) saying 'Monster Mash'. The user will be able to click on the banner and be taken to the homepage.

#### 2.3.1.6. Homepage



Any open offers the user has such as

friends requests will appear here. From

here, the user can accept/reject the offer.

Messages from other friends will also

appear here.

The user's monsters will appear here and will

have the monster's stats along with value.

It will also show if the monster is up for

sale or breeding. From this, a friend can

visit this stable and accept to breed/buy.

Each friend of the user will

appear here, along with their

cash amount. This would be

in order of the wealthiest.

#### 2.3.1.7. Friend's Stable

This is the user's sign up email.

The friend's monsters will appear next to

each other here, and will show the monster

name, attributes, value and give you the option

To challenge them to a battle, request to

breed and put in a bid to buy that monster.

**1**

**3**

**2**

Each player has their own cash pile,

which shows how much virtual

money they have. The friend's amount

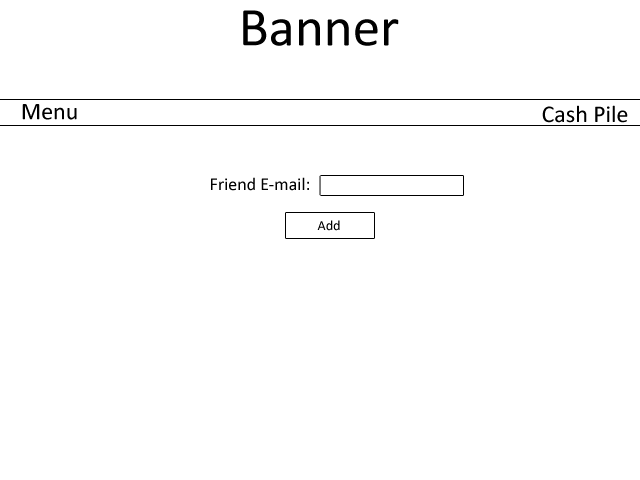
of virtual money will appear here.

1. User can challenge this friend's monster

2. User can request to breed with this monster

3. User can put a bid in to buy this monster

#### 2.3.1.8. Add Friend



If the e-mail is a valid address,

the friend will be sent a request from

this user and will be seen in the

offers section on the homepage.

If the e-mail is not recognized, an

error message will appear.

The e-mail that the friend used to

sign up will be used for user's

to add them.

#### 2.3.1.9. Shop



When a user loses all their monsters, they

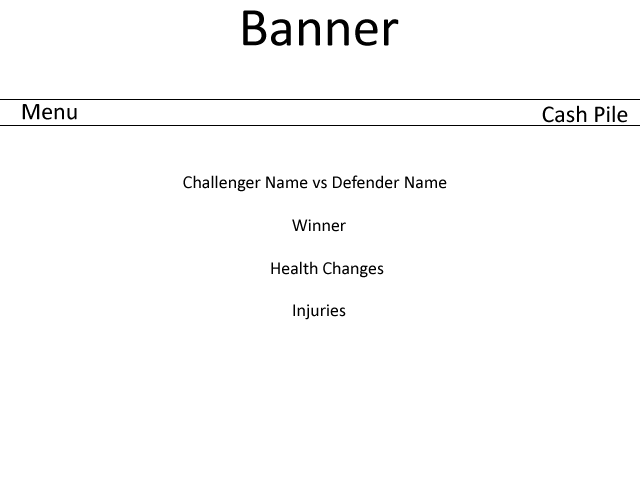
can buy another monster from this page. After

clicking buy, the money will be deducted from

their cash pile and the monster

will then be their own.

#### 2.3.1.10. Battle Report



Any injuries the monster gets will

be stated here. If the monster dies,

an image will appear of a sad monster

with a link to the “shop” page

to buy another monster.

After each battle, an (not yet specified)

amount will be deducted from

the monster's health. This will show

how much health it has lost and the

monster's current health (including

the deduction).

The winner of the match will be here.

The challenger will be the person that sent the request.

The defender will be the person that accepted it.

## 2.5. Risk Assessment Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Risk Severity** | **What Will It Affect?** | **Measures Put In Place** | **Other Notes** |
| Illness | Medium-High | Project milestones and group centralisation. | Alert group as soon as possible so group Could act accordingly. |  |
| Natural Disaster | Low-High | Depending on the natural disaster, progress may be affected in different ways. e.g. flooding may change team members’ priorities. | Contact numbers distributed to group and try to a communication channel open with group to deal with problems. |  |
| Small coding error and unable to track source of error | Low | Ability to proceed with implementation.  Increased stress to | Use version management to roll back code to last working commit of code. |  |
| Deletion of local git repository | Low | Could affect progress of coding/ loss of code. | Commit updates frequently to avoid losing too much code.  Can re-clone online repository to local system. |  |
| Deletion of online git repository | Medium-High | Could lose whole project work. | Make sure each group member has an up-to-date local clone of the git repository to re-upload to the online repository/recreate a new one. | To delete the whole online git repository, the version management controller must delete it and manually and confirm deletion by entering the name of the repository to be deleted. |
| Code incompatibility | Low-High | Interaction of code between group members’ work could be hit and cause program-wide problems. | Make sure coders meet up frequently and work on code together along with QA manager.  Keep all work as centralised as possible by having frequent group meetings in which to do work/assign task at. | If work becomes decentralised, code incompatibility could become a big problem. |
| Server-server interaction problems | Medium-High | Servers may not be able to communicate with each other. | Make sure frequent meetings between allocated members from other groups are arranged to discuss server-server interaction protocols. | Keep the program as simple as possible but making sure that the program meets all of the requirements.  By keeping it as simple as possible, it is less likely that server-server interaction problems will occur. |
| Loss of project direction | Medium | Wrong tasks being allocated so wrong work is produced for delivery. | Frequent group meetings, checking requirements specification and appropriate documents to find check if the right goals are being worked towards at the right time. |  |
| Individual circumstances | Low-High | Could affect work motivation/priorities as well as group dynamic depending on situation. | Group supports each other appropriately having meetings to decide what to do if needed. Handle delicately. | May not become aware of individual’s circumstances straight away but this is expected. |
| Browser compatibility | Low | Client requires program to run on all installed browsers in the Delphinium and Solarium. | Keep interface with browser simple and validate to make sure it is compatible. |  |

## 2.6. Task Allocation

Due to difficulty in presenting the document in a readable state, see the **“Project Plan Task Allocation.pdf”** document for the task allocation. This will be sent along with the Project Plan and the Gantt Chart.

## 2.7. Gantt Chart

For the same reasons that the task allocation table cannot be presented as a readable fashion, see the **“Project Gantt Chart 27-10-2012.pdf”** document for the Gantt Chart. This will be sent along with the Project Plan and the Task Allocation Table.

# 3. References

N/A

# 4. Document Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF No. | Date | Changes made to document | Changed By |
| 1 | N/A | 26/10/12 | This is the first draft | All (each member wrote their part) |
| 1.1 | N/A | 26/10/12 | Table of contents, Introduction and formatting was added. | sjm16 |
| 1.2 | N/A | 27/10/12 | Adding of risk assessment table and note of Task Allocation and Gantt Chart will be sent along with this Project Plan. | sjm16 |
| 1.3 | N/A | 09/12/12 | Removal of Battle Report Seq. Diagram | sjm16 |