Software Engineering Group Project

Final Documentation

|  |  |
| --- | --- |
| Author: | Joshua Bird, Phil Wilkinson, Thomas Hull, David Haenze, Christian Morgan, Kamarus Alimin, Szymon Stec, Lewis Waldron |
| Config Ref: | SE\_02\_FC\_01 |
| Date: | 2010-10-11 |
| Version: | 1.0 |
| Status: | Final |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2012

CONTENTS

CONTENTS 2

1. Introduction 3

2. Purpose of this Document 3

2.2 Objectives 3

3. BODY OF DOCUMENT 3

REFERENCES 3

DOCUMENT HISTORY 3

Software Engineering Group Project

Final Documentation

|  |  |
| --- | --- |
| Author: | Joshua Bird, Phil Wilkinson, Thomas Hull, David Haenze, Christian Morgan, Kamarus Alimin, Szymon Stec, Lewis Waldron |
| Config Ref: | SE\_02\_FC\_01 |
| Date: | 17-02-2013 |
| Version: | 1.0 |
| Status: | Final |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2013

CONTENTS

CONTENTS 4

1. Introduction 5

1.1 Purpose of this Document 5

1.2 Scope 5

1.3 Objectives 5

2. Project Plan 6

2.1 Overview 6

2.2 Use Case Diagram 7

2.3 User interface design 10

2.4 Gantt chart 14

2.5 Risk Assessment 1

REFERENCES 3

DOCUMENT HISTORY 4

# Introduction

## Purpose of this Document

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

## 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.

## 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.

# Project Plan

## 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.

## Use Case Diagram

### Use case friend diagram

-Sell Monster

-Breeding Monster

-Buy Monster

-Challenge Monster

-Accept Monster Challenge

-View Friend Wealth

-Delete Friend

Friend

User

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.

### Use Case Non Friend Diagram

Non Friend

User

-Send Friend Request

-Accept Friend Request

-Decline Friend Request

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.

### Use Case Server Diagram

-Register

-Un-register

-Log in

-Log Out

-View Rich List

-Maintain Friend/Monster Lists

-Display Friend List

-Decide Fight Outcome

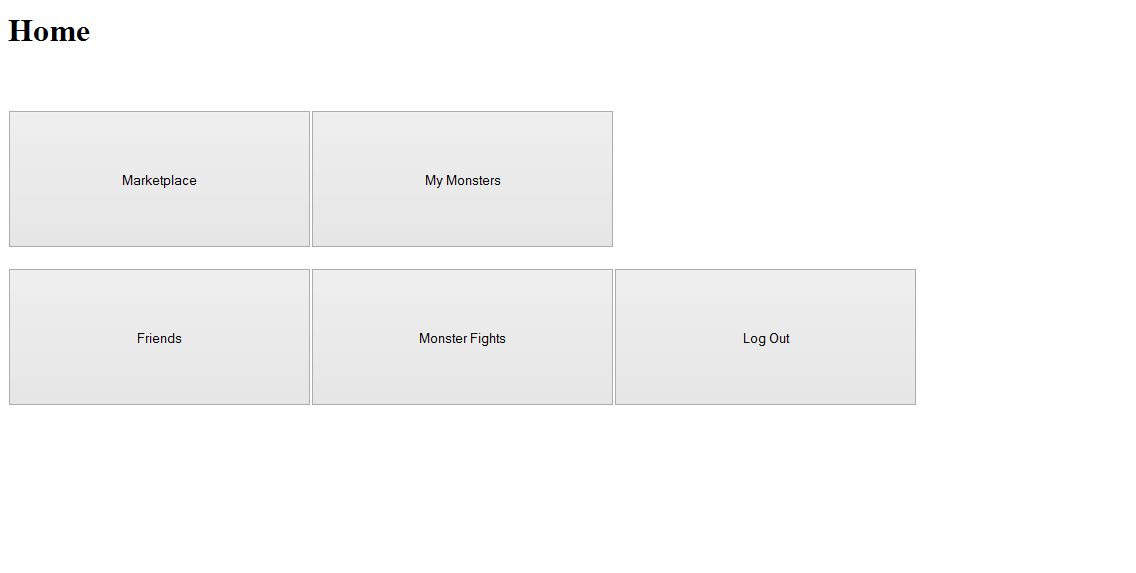
User

Server

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.

## User interface design

### 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)

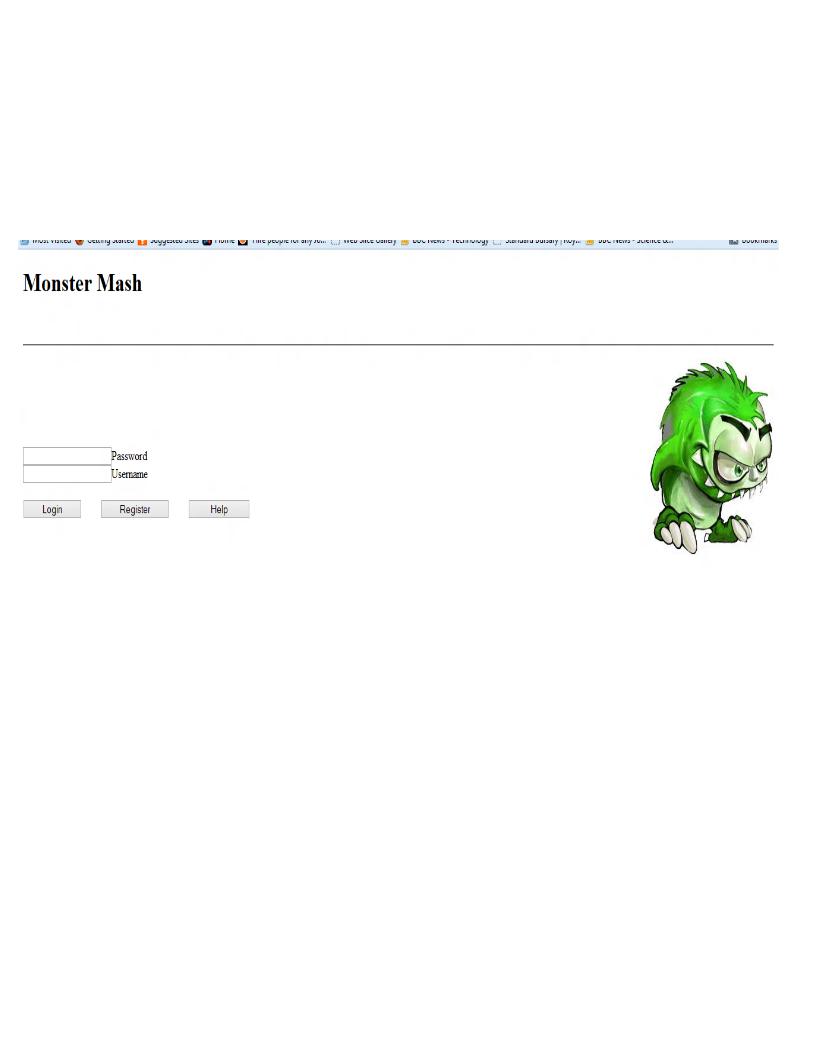
My Monsters > myMonsters.html (where one can breed, sell and loan monsters)

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)

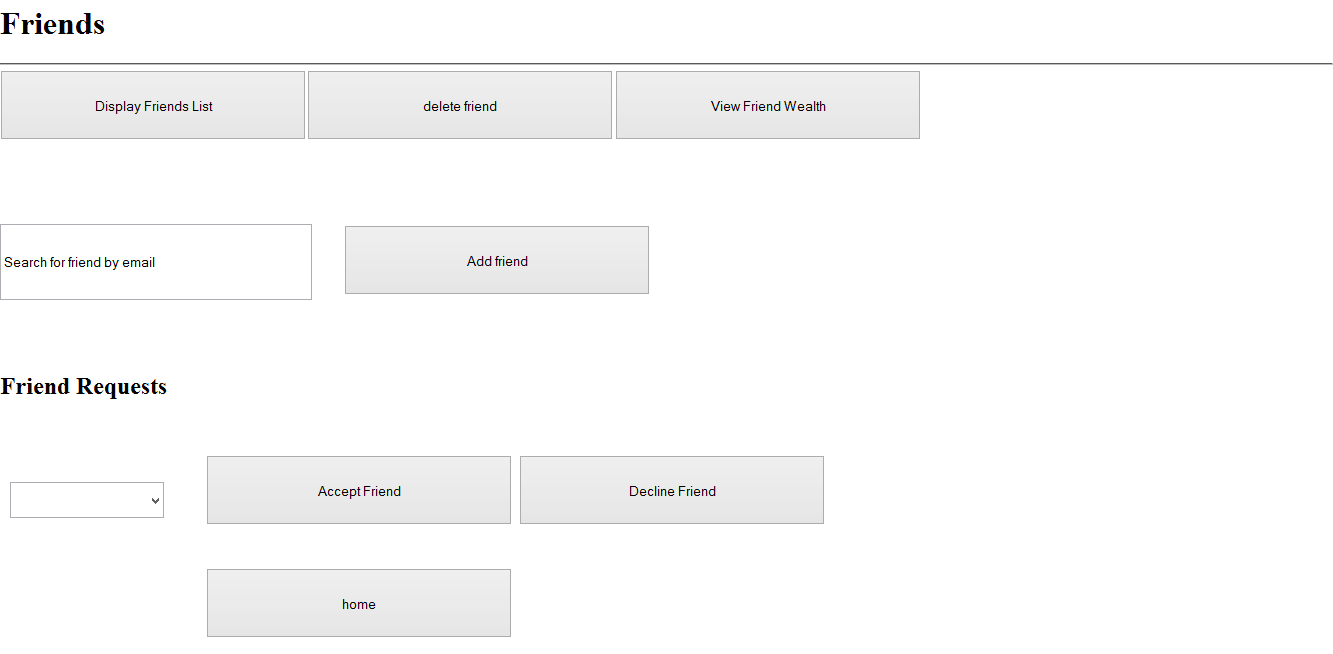
Log Out > login.html (Takes the user back to the log in page.)

### Login screen



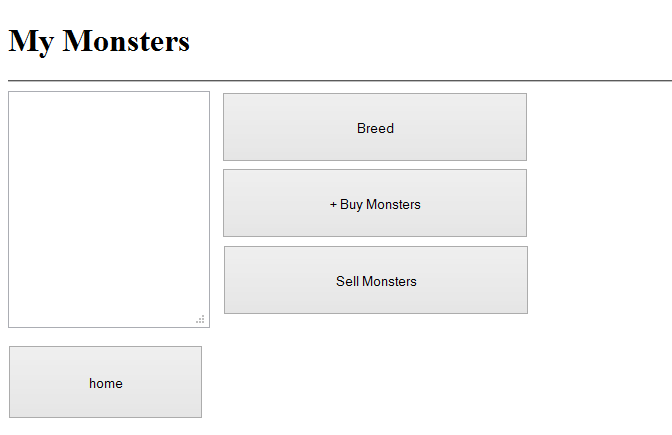
When the game is started the first page the user should see is the log-in page. As the user can see here, one can log-in with their correct credentials. If the user were not to have any, they can register (or view the help page).

### 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.

### List of user’s current monsters



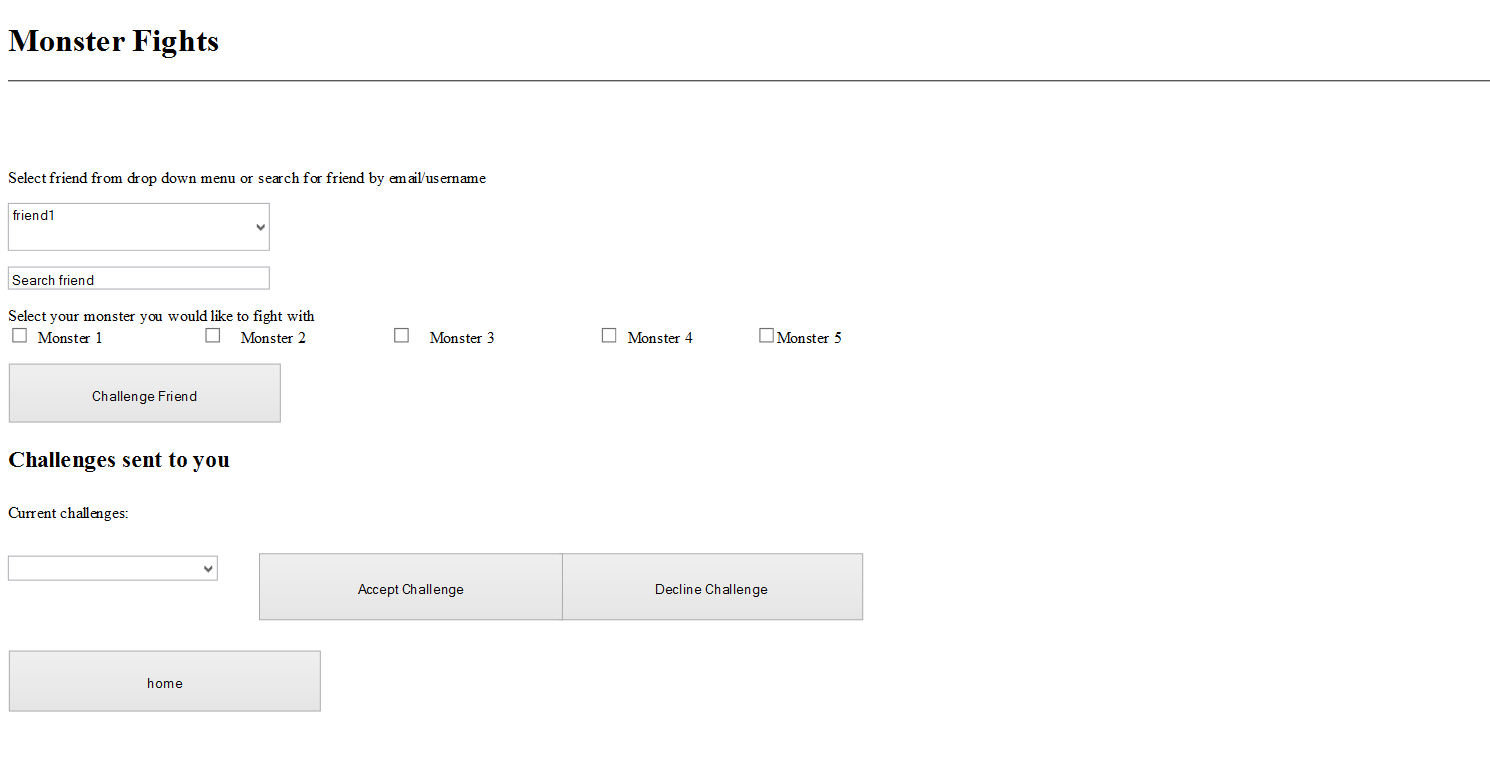
After clicking the ‘my monsters’ button on the home page the following page will be shown. A user will be able to breed a monster, as when the button is pressed, a prompt box with a drop-down box of your monsters will be shown, enabling the user to choose which monsters to breed. The ‘buy monsters’ button 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.

### Buy a new monster



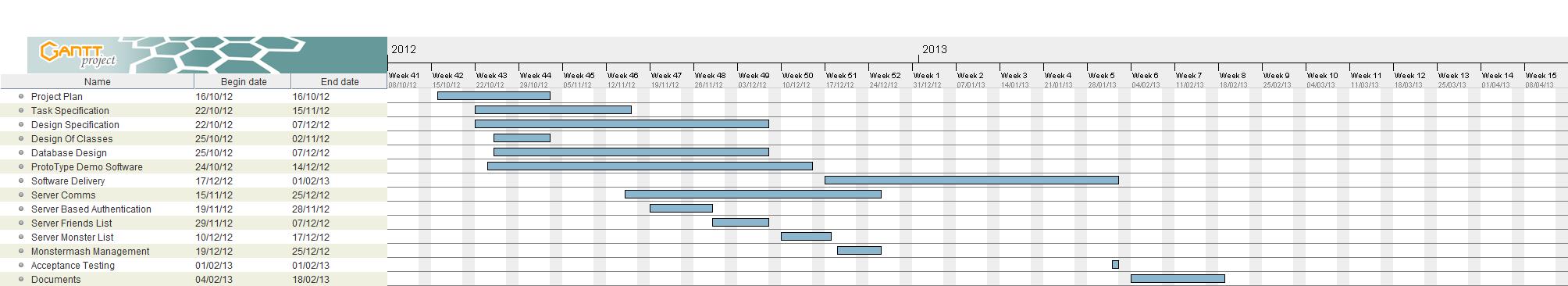
After clicking the marketplace button on the home page this page shall load. There is a list box on the left-hand side which will display monsters that other friends are selling or loaning out. After clicking on the monster, the user shall be able to rent or buy them by clicking the appropriate buttons. 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.

### 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.

## Gantt chart



## 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 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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 | 12/09/08 | N/A - original version | CJP |
| 1.1 | N/A | 2010-10-12 | Updated to Office 2007; key field automation increased; paragraph styles updated | NWH |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Software Engineering Group Project

Final Documentation

|  |  |
| --- | --- |
| Author: | Joshua Bird, Phil Wilkinson, Thomas Hull, David Haenze, Christian Morgan, Kamarus Alimin, Szymon Stec, Lewis Waldron |
| Config Ref: | SE\_02\_FC\_01 |
| Date: | 14-11-2012 |
| Version: | 1.0 |
| Status: | Final |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2012

CONTENTS

CONTENTS 2

1. Introduction 3

1.1 Purpose of this Document 3

1.2 Scope 3

1.3 Objectives 3

2. Approach to testing 3

3. Test Specification 4

REFERENCES 8

DOCUMENT HISTORY 9

# Introduction

## Purpose of this Document

The purpose of this document is to outline how the design specification/program will be tested against the tests in the test specification table.

## Scope

The test plan aims to set standardisation for the testing of the program, and parameters in which the program will be tested to. All program testers are required to read this document to understand the guidelines they will follow when testing the program.

## Objectives

The objectives of the test plan are to see whether the features in the program work, given different variables. The testing will be done in the following way:-

* the first column will contain the test reference;
* the second column will contain the requisite being tested;
* the third column is test content which will give a short description of what is being tested;
* the fourth column will contain the input, i.e. what action will be done;
* the fifth column will contain the output after the action has taken place;
* the sixth column will contain the pass criteria i.e. how the test will be passed.
* the seventh column will contain the Test criteria i.e. to determine if that each test cases is a success or failure

# Approach to testing

Testing is a fundamental component of the software quality assurance and represented a review to meet the requirement as specified in the document QA Document SE.QA.01. The technique used to test is the black box testing techniques, where we perform functional testing to ensure that the functionality meets the requirements. In this approach, we focus on determining whether or not a program does what it is supposed to do based upon its functional requirements. Black box testing attempts to find errors in the external behaviour of the code (output) which is grouped into the following categories ; incorrect or missing functionality; interface errors; errors in data structures used by interfaces; behaviour or performance errors; and initialization and termination errors. Through this testing, we can determine if the functions appear to work according to specifications. The main black box testing tools used are the record and playback tools - the IDE used to test the system against these requirements is Netbeans. Black-box testing focusses on the inputs and outputs of the software without knowing their internal code implementation .The following is the guideline that should be followed during the testing phase.

1. Conform to the requirements and specifications that is shown and set in the QA documents
2. Conduct a number of test cases by choosing valid input to check the positive test scenario and invalid input to check the negative test scenario of the system
3. Tester should determine the expected output from the input given.
4. Using the specified inputs, check that they match the expected output/result
5. Chosen test cases are executed and documented for the outcome
6. If any errors occur, it should be fixed if possible
7. Test table is drawn up to show the number of tests conducted and whether they passed or failed.

# Test Specification

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Ref** | **Req. being tested** | **Test content** | **Input** | **Output** | **Pass Criteria** | **Test pass/fail?** |
| SE-N02-001 | FR1 + FR6(register) | Test whether the “Register button” on the login page will navigate the user to the register page | Click the register button first - the button on the left on the login screen | A page which consists of forms and buttons will be shown on the user’s browser | The page should be load correctly | Pass |
| SE-N02-002 | FR1 + FR6(register) | Check if the system does not allow adding already existing user | Enter an user email that already exists in database | Account is not created, user gets error message | System displays  "User already exists"  error message | Pass |
| SE-N02-003 | FR1 + FR6 (Register) | Check if the system does not allow adding new user if all required fields are not filled | Leave all the fields blank | Account is not created, user gets error message | System displays  "You have to fill all required fields"  error message | Pass |
| SE-N02-004 | FR1 + FR6 (Register) | Check if the system does not allow using restricted characters | Enter special characters (^%#) in email section | Account is not created, user gets error message | System displays  "Incorrect input"  error message | Pass |
| SE-N02-005 | FR1 + FR6 (Register) | Check if system creates new user account | Enter user email and all required data correctly | List of stored users should now include this user. | Data should be stored  correctly | Pass |
| SE-N02-006 | FR1 + FR6 (Unregister) | Check if system deletes user data from database | Logged user clicks “Unregister” Option | List of stored  users should not include this user. User is redirected to login page | Data should be remove correctly | Fail |
| SE-N02-007 | FR6 (Unregister) | Check if system deletes user data from database | Logged user clicks “Unregister”  Option | Could not remove the user | System displays  "Some problems occurred, please try again or contact administrator " error message | Fail |
| SE-N02-008 | FR2 + FR6 | Check if the “friends button” on the Home page navigates the user to the ‘friends.html’ page | Click the friend button on the home page | The ‘friends.html’ page will be shown on the user’s browser | The page should be load correctly | Pass |
| SE-N02-009 | FR2 + FR6 (Add Friend) | Check if system does not allow to add non existing user | Enter user email that does not exist in database  or leave it blank | Friend should not be added | System displays “Sorry, user does not exist” error message | Pass |
| SE-N02-010 | FR2 + FR6 (Add Friend) | Check if user is added correctly if request accepted | Enter friends email correctly and person accepts friend request | Friend should be added correctly | Friend is added to user’s friends list | Pass |
| SE-N02-011 | FR2 + FR6 (Add Friend) | From the test SE-N02-012, has the other user been notified with the respective friend request | Log-on to the other account which is about to receive the friend request | There should be a notification in the friend list page with the details of the email of the user. | The friend request should perform well | Pass |
| SE-N02-013 | FR2 + FR6 (Add Friend) | From test SE-N02-014, we check if the user is able to successfully send two friend requests. | Send the two friend requests by using their email in the friend request page. After that, login to that chosen email account | Both account should receive the friend request | Adding more than one friend request in the friend page is successful | Pass |
| SE-N02-015 | FR2 + FR6 (Decline friend request) | To test the decline button of the friend request session | From the test SE-N02-010 ,we test it out with the decline button | If the person is declined, that user will not appear on the friend request box. | Friend invitation should be performing well | Pass |
| SE-N02-016 | FR2 + FR6 (Remove Friend) | Check if system allows removing friend without selecting any from the list | No friend is selected, use delete button | Error message should be displayed | System displays “You have to select a friend first.” error message | Fail |
| SE-N02-017 | FR2 + FR6 (Remove Friend) | Check if friend is deleted correctly | Select friend from “friend list”, use delete button | Friend should be deleted correctly | Friend is deleted from user’s friends list | Fail    Delete Friend function is unavailable |
| SE-N02-018 | FR3 + FR6 | Check if the “monster button” on the homepage will navigate the user to the monster menu page | Click the “monster button” shown in the homepage | The browser loads myMonster.html onto the screen | The navigation button should direct the user to the selected page | Pass |
| SE-N02-019 | FR3 (New account) | To verify if a Newbie package such as virtual money, is given to new users after their registration | Create an account on the login screen | After creating an account, the system will automatically allocate virtual money to the new account | Newbie package system should works | Pass |
| SE-N02-020 | FR4 | Test whether the user monster can challenge another monster | We test out by using an admin account: search another valid account or search from the list of friends | The chosen user will appear from the drop-down menu | The monster challenge system should works | Pass |
| SE-N02-021 | FR4 | Check if the user can accept a challenge from another user | Use an account to invite another member to a challenge | There will be a notification from the homepage that another user wishes to challenge | Challenge system should works | Pass |
| SE-N02-022 | FR4 | Check if the user can decline a challenge from another user | Use an account to invite another member to a challenge and have that other member decline the challenge | There will be a notification from the homepage that other user declined the challenge | Declining challenge from the user should works | Pass |
| SE-N02-023 | FR4 | To confirm that the winner of the monster challenge receives their reward | From test SE-N02-017, which is another account accepts the challenge from the current account to test which has a more powerful monster | There will be a short video play on screen of monsters fighting while the system calculates the winner | The monster challenge should works | Fail  No media is loaded |
| SE-N02-024 | FR6 (Offer for sale) | Check if system allows selling a monster without selecting any from the list | No monster is selected, use sell monster button | Error message should be displayed | System displays “You have to select a monster first.” error message | Pass |
| SE-N02-025 | FR6 (Offer for sale) | Check if system allows monsters to be sold | Select monster from “monster list”, use sell monster button | Monster should be added to the market list | Monster is added correctly to the market list | Pass |
| SE-N02-026 | FR6 (Offer for sale) | Check if system allows monsters with no price to be sold | Set monster price to 0 or leave it blank | Monster is not added to the market list, error message should be displayed | System displays “You have to set a monster price first” error message | Fail |
| SE-N02-027 | FR6 (Buy monster) | Check if system allows user with not enough funds to buy a monster | Select monster in market, use buy button | Monster is not added to users monsters list | System displays “You don’t have enough money to buy this monster” error message | Fail |
| SE-N02-028 | FR6 (Buy monster) | Check if system allows buying a monster without selecting any from the market list | Select no monster in market, use buy it button | No monster should be added to users monsters list | System displays “You have to select a monster first” error message | Fail |
| SE-N02-029 | FR6 (Buy monster) | Check if system allow to buy a monster from other user | Select monster, use buy button on the user | Monster should be added to users monsters list and removed from market | Monster is correctly added to users monsters list and removed from market | Pass |
| SE-N02-030 | FR6 (Offer for breeding) | Check if system allows breed a monster without selecting any from the list | No monster is selected, use breed monster button | Error message should be displayed | System displays “You have to select a monster first.” error message | Fail |
| SE-N02-031 | FR6 (Offer for breeding) | Check if system allows monsters with no price to be bred | Set breed price to 0 or leave it blank | Monster is not added to the market list, error message should be displayed | System displays “You have to set a breed price first” error message | Pass |
| SE-N02-032 | FR6 (Offer for breeding) | Check if system allows monsters to be bred | Select monster from “monster list”, use breed monster button | Monster should be added to the market list | Monster is added correctly to the market list | Pass |
| SE-N02-033 | FR6 (Purchase breeding) | Check if system allows user with not enough funds to buy a monster | Select monster in market, use breed button | Monster is not added to users monsters list | System displays “You don’t have enough money to buy this monster” error message | Pass |
| SE-N02-034 | FR6 (Purchase breeding) | Check if system allows buying a monster without selecting any from the market list | Select no monster in market, use breed button | No monster should be added to users monsters list | System displays “You have to select a monster first” error message | Fail |
| SE-N02-035 | FR6 (Purchase breeding) | Check if system allow to buy a monster for breeding | Select monster, use breed button | New monster should be added to users monsters list | New monster is correctly added to users monsters list | Fail |
| SE-N02-036 | FR1 + FR7(Log-in) | Test whether the ”login button” on the login page works – this will navigate the user to the home page | Entering a valid username with the password and clicking the login button | After the login process, the home page will be loaded onto the user’s browser | The page should have been load correctly | Pass |
| SE-N02-037 | FR7(Log-in) | Check if the system allows to log-in not providing any details | Leave blank user log-in and password | Error message should be displayed | System displays “Please, enter your username and password.” error message | Pass |
| SE-N02-038 | FR7(Log-in) | Check if the system allows not existing users to log-in | Enter new user email and password | User is not logged-in, error message should be displayed | System displays “Wrong username or/and password” error message | Pass |
| SE-N02-039 | FR7(Log-in) | Check if the system allows user to log-in no password | Enter only username, leave password blank | User is not logged-in, error message should be displayed | System displays “Wrong username or/and password” error message | Pass |
| SE-N02-040 | FR7(Log-in) | Check if the system allows user to log-in with incorrect password | Enter username and wrong password | User is not logged-in, error message should be displayed | System displays “Wrong username or/and password” error message | Pass |
| SE-N02-041 | FR7(Log-in) | Check if the system allows user to log-in | Enter username and password | User should be logged-in | User is logged-in and redirected to the home page | Pass |
| SE-N02-042 | FR7(Log-out) | Check if the systems allows user to log-out | Use log-out button | User should be logged-out | User is logged out and redirected to the log-in page | Pass |
| SE-N02-043 | FR8(monsters list) | Check if the system loads the list of user’s monsters correctly even though it is empty | Use “my monsters“ button | System should show an empty list | System should be loaded an empty list correctly | Pass |
| SE-N02-044 | FR8(monsters list) | Check if the system loads the list of user’s monsters | Use “my monsters“ button | System should show a list of monsters | System should be loading the list correctly | Pass |
| SE-N02-045 | FR8(friends list) | Check if the system loads the list of user’s friends correctly even though it is empty | Use “friends“ button | System should show an empty list | System should be loading an empty list correctly | Pass |

REFERENCES

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

[2] Software Engineering Group Projects: Test Procedure Standards. C. J. Price, N. W. Hardy. SE.QA.06. 1.6 Release

DOCUMENT HISTORY

| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
| --- | --- | --- | --- | --- |
| 1.0 | N/A | 14/11/2012 | N/A - original version | PW |
| 1.1 | N/A | 09/02/2013 | Approach to testing created and tests fixed | KA |
| 1.2 | N/A | 10/02/2013 | Fixed spelling/grammatical mistakes | PW |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |