**Software Engineering Group Project 15**

**Final Submission**

**Authors: Samuel Sherar,**

**Kamil Mrowiec**

**Fiona Joseph Samy**

**Andy Watkins**

**Imran Mungul**

**Grand David**

**Cellen Williams**

Date: 14/02/2013

Version: 1.0

Status: Draft

Table of Contents

Requirements and Project Plan Document 4

Purpose of this Document 4

Scope 4

Objectives 4

Overview 4

Use Case Diagrams 6

User Interface Design 7

Gantt Chart 11

Risk Analysis 14

Change log 15

Testing Document 16

1. INTRODUCTION 16

1.1 Purpose of this Document 16

1.2 Scope 16

1.3 Objectives 16

2. TEST SPECIFICATION 16

3. REFERENCES 21

4. DOCUMENT CHANGE HISTORY 21

Design Document 22

Introduction 22

Purpose of this Document 22

Scope 22

Objectives 22

Decomposition Description 23

Class Diagram 23

Class Description 26

Mapping from requirements to Classes 27

Dependency Description 28

Component Diagrams 28

Detailed Design 29

Sequence Diagrams 29

Change log 35

Completed Testing Document 36

Final Report 41

Management Summary 41

Timeline of the Project 42

Requirements Document 42

Testing Document 42

Design Document 42

Integration and Testing Week 42

Final Report 43

Final State of the Project 43

Performance of Members 43

Sam Sherar <sbs1> - Team Leader 43

Kamil Mrowiec <kam20> - Co-Team Leader 43

Fiona Joseph Samy <fij1> - QA Manager 43

Imran Mungul <Imm5> - Design Team 44

Grant David <grd3> - Testing Team 44

Cellen Williams <cew10> - Testing Team 44

Time Sheets 44

Sam Sherar <sbs1> 44

Kamil Mrowiec <kam20> 44

Fiona Joseph Samy <fij1> 45

Andrew Watkins <ajw14> 45

Grant David <grd3> 45

Cellen Williams <cew10> 46

Imran Mungul 47

Personal Reviews 47

Sam Sherar <sbs1> 47

Kamil Mrowiec <kam20> 48

Fiona Joseph Samy <fij1> 48

Imran Mungel <imm5> 48

Andrew Watkins <ajw14> 49

Grant David <grd3> 50

Cellen Williams <cew10> 50

Change log 51

# Requirements and Project Plan Document

## Purpose of this Document

The purpose of this document is to clarify and define the project mandate. It will state the possible constraints that the group could face in regards to producing the Monster Mash game. The processes the group will have to commit will also be defined. This document will have to stay within the requirements and specification of the QA documents.

## Scope

In this document, we will go over the plan for the project, on both a time and user perspective. We have included basic user interaction design, including use case diagrams, as well as decisions we have made in regards to the project.

## Objectives

This document’s objective is to show the reader what materials we used to produce our Monster mash game.

* Use-case diagrams
* User interface design
* Gantt chart
* Risk Analysis

All of the above are material that will be explained in this document with the necessary screenshots and information stated.

## Overview

For the project we have decided to use Glassfish for the server interaction as it is more flexible compared to Google App Engine (GAE) in regards to data persistence and it means that the end users don’t have to register for a Google account when authenticating with our application.

For the Version Control System, we have gone with Git over SVN, as one of our developers know how to use it and is able to teach the rest of the group how to use. Our repository is hosted on [www.github.com](http://www.github.com) under a private account, so we can control who can access it.

We are using Microsoft Word for documentation and Microsoft Visio / Lucid Chart for diagrams.

## Use Case Diagrams

# 

Here is the Use-Case diagram, clearly showing what the user and system should be able to do. The user needs to be able to:

* Register or log in
* Host a monster farm
* Set monster attributes
* Interact with other players
* Breed, fight and purchase monsters
* Change their password

The system needs to be able to:

* Display:
  + A list of monsters and attributes
  + A list of friends
  + Friend requests
* Communicate with other servers
* Store details of users and monsters
* Allow users to register
* Maintain and manage friend lists
* Manage monsters lifecycle

## User Interface Design

Fig.1 - Main page

This is our main page for the project. Here we have 4 links: Homepage, Login/Register/Logout, Fight and Stable. When the user comes to the main page, to begin with they only can interact with two of the links (Homepage and Register/login). When the user has been authenticated Register/login changes to logout, and two links appear to allow the user to fight monsters and view the stable.

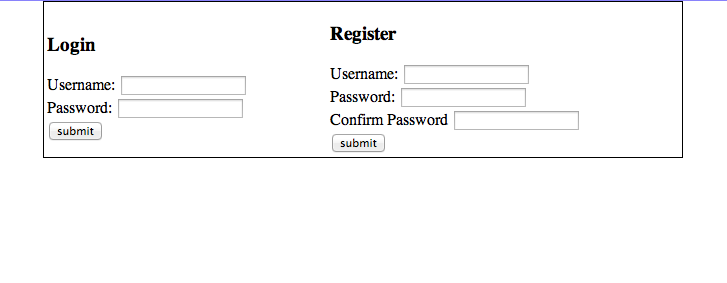


Fig 2 - Registering and Logging in

Here is the form for the user to both login and register. It is clearly marked out so the user can distinguish between both forms.

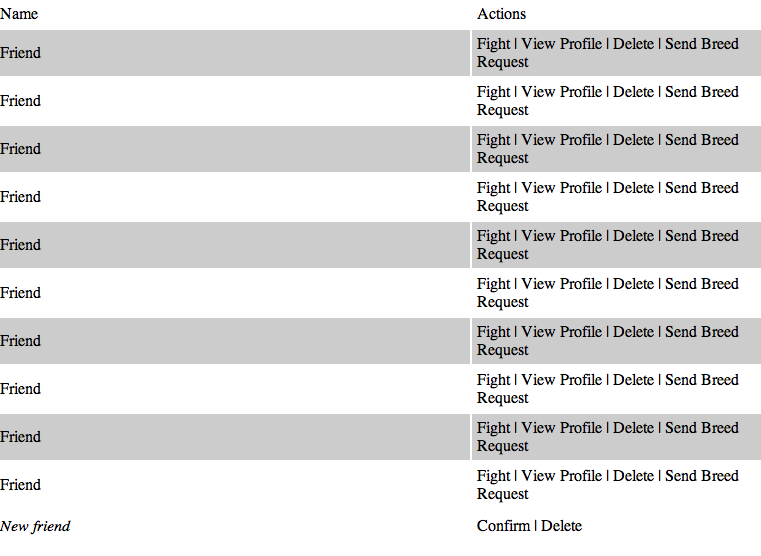


Fig 3 - Friends list

Here is the list of friends which the user may have - both confirmed and requests. With confirmed friends, there are more interactions available, such as fighting and sending breed requests. Unconfirmed friends are in italics at the bottom with only an option to confirm or delete.

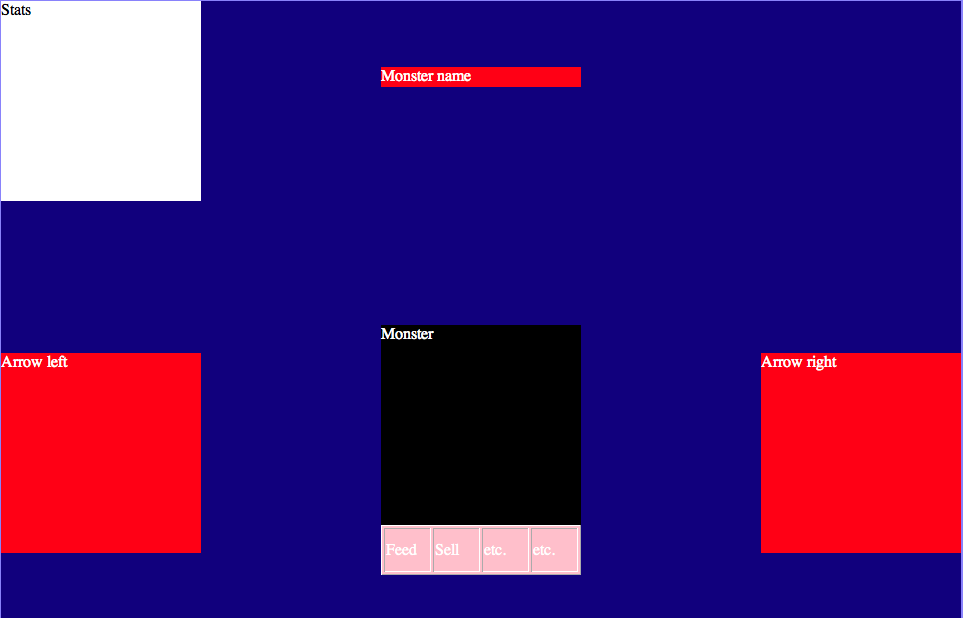


Fig 4 - Monster Information

We will be designing the stable and monster information into more of a console game, so we have the ability to scroll 1 by 1 back and forward between all of our monster. We will also give access to a list. There will be certain actions the user can do with the monster, such as feed and sell.

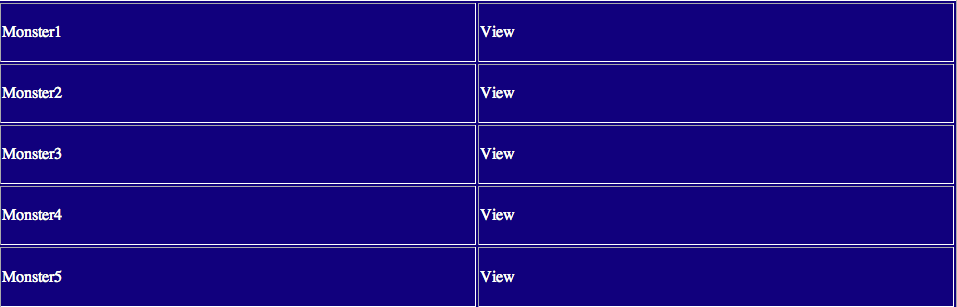


Fig 5 - Stable list

As well as the monster information screen, we have a list with certain quick actions on there, such as view the monster.

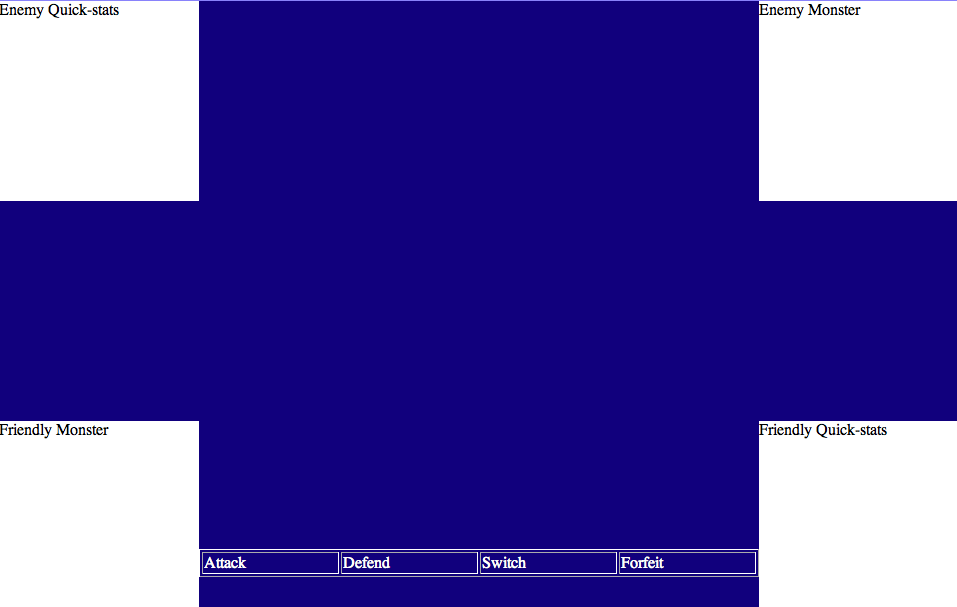
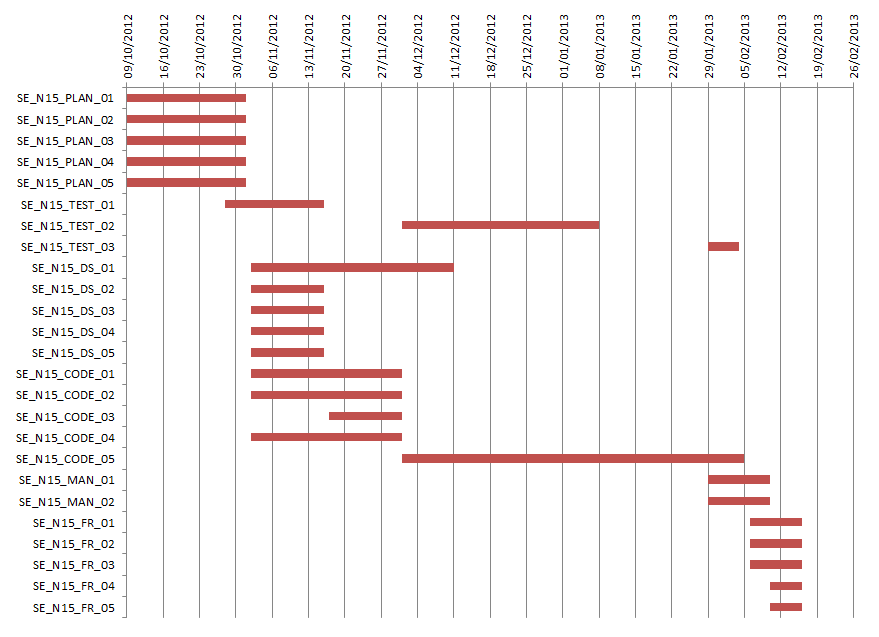


Fig 6 - Fight screen

We have created the interface as simply as we can, with only a certain amount of actions the user can perform during the fight.

## Gantt Chart

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Predecessors | Task Owner | Start Date | End Date | Description | Additional members |
| SE\_N15\_PLAN\_01 | PLAN\_02 - 05 | Sam Sherar | 09/10/2012 | 01/11/2012 | Writing the Overall Document |  |
| SE\_N15\_PLAN\_02 |  | Cellan Williams | 09/10/2012 | 01/11/2012 | Creating Use Case diagrams |  |
| SE\_N15\_PLAN\_03 |  | Andy Watkins | 09/10/2012 | 01/11/2012 | User Interaction design |  |
| SE\_N15\_PLAN\_04 |  | Charlie Bird | 09/10/2012 | 01/11/2012 | Risk Analysis |  |
| SE\_N15\_PLAN\_05 |  | Sam Sherar | 09/10/2012 | 01/11/2012 | Gantt Chart |  |
| SE\_N15\_TEST\_01 |  | Grant David | 28/10/2012 | 16/11/2012 | Writing the Test Document |  |
| SE\_N15\_TEST\_02 | TEST\_01 | Grant David | 01/12/2012 | 08/01/2013 | Testing Prototypes | Imran Mungul |
| SE\_N15\_TEST\_03 | TEST\_02 | Grant David | 29/01/2013 | 04/02/2013 | Testing Final Product | Imran Mungul |
| SE\_N15\_DS\_01 | DS\_02 - 05 | Charlie Bird | 02/11/2012 | 11/12/2012 | Writing the Overall Document |  |
| SE\_N15\_DS\_02 |  | Imran Mungul | 02/11/2012 | 16/11/2012 | Decomposition Description |  |
| SE\_N15\_DS\_03 |  | Andy Watkins | 02/11/2012 | 16/11/2012 | Dependancy Description |  |
| SE\_N15\_DS\_04 |  | Cellan Williams | 02/11/2012 | 16/11/2012 | Interface Description |  |
| SE\_N15\_DS\_05 |  | Fiona Samy | 02/11/2012 | 16/11/2012 | Detailed Design |  |
| SE\_N15\_CODE\_01 | CODE\_02 - 04 | Kamil Mrowic | 02/11/2012 | 01/12/2012 | Prototypes |  |
| SE\_N15\_CODE\_02 |  | Kamil Mrowic | 02/11/2012 | 01/12/2012 | Server-Server Interaction |  |
| SE\_N15\_CODE\_03 |  | Andy Watkins | 17/11/2012 | 01/12/2012 | User Interface |  |
| SE\_N15\_CODE\_04 |  | Sam Sherar | 02/11/2012 | 01/12/2012 | Data Persistance and Modelling |  |
| SE\_N15\_CODE\_05 | CODE\_01 | Kamil Mrowic | 01/12/2012 | 05/02/2013 | Developing Final Product | Sam Sherar, Andy Watkins, Cellan Williams |
| SE\_N15\_MAN\_01 |  | Fiona Samy | 29/01/2013 | 10/02/2013 | Writing User Manual |  |
| SE\_N15\_MAN\_02 |  | Imran Mungul | 29/01/2013 | 10/02/2013 | Writing Technical Manual |  |
| SE\_N15\_FR\_01 |  | Sam Sherar | 06/02/2013 | 16/02/2013 | Writing Final Report | Everyone |
| SE\_N15\_FR\_02 | PLAN | Andy Watkins | 06/02/2013 | 16/02/2013 | Finalising Project Plan document |  |
| SE\_N15\_FR\_03 | TEST | Grant David | 06/02/2013 | 16/02/2013 | Finalising Test Document |  |
| SE\_N15\_FR\_04 | MAN | Fiona Samy | 10/02/2013 | 16/02/2013 | Finalising User manual |  |
| SE\_N15\_FR\_05 | DS | Imran Mungul | 10/02/2013 | 16/02/2013 | Finalising Design Specification document |  |



## Risk Analysis

1. Group-to-Group coordination – if not organised and handled correctly this could easily

generate a lot of wasted time. If groups don't agree on common protocols it could become

very awkward to program the server-to-server part of the project. This should be allowed for

when time is being allocated.

2. Slippage – If slippage caused by certain parts of the project occurs then other members should be drafted in to help speed up that part of the project.

3. Illness – If a group member is ill, they should notify the project leader to have their current

assignment reassigned, if possible. If not then they should be included in other assignments

once they are well again, in order to make up for the lost time.

4. Complicated algorithms – if the individual assigned to create an algorithm is struggling with

it, more members of the group can be assigned to reduce time loss.

5. Authentication – should be kept separate from the other group projects in order to maintain

security.

6. User interface – needs to be suitable for a primary/secondary school audience, meaning it

must be simple to understand and use. If the audience can't understand how to use it, it's

unlikely to be popular.

7. Git – Whilst Git is a very useful program, it is very important that space is managed, as

everyone having a copy of the entire repository could take up quite a lot of space.

8. Data Protection – as the application will is aimed at a school audience it is especially

important that data is kept secure, especially if they are to input personal details.

## Change log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF Number | Date | Changes | User |
| V1.0 | N/A | 23/10/12 | First draft of the document | sbs1 |
| V1.1 | N/A | 25/10/2012 | User Interface | ajw14 |
| V1.2 | N/A | 26/10/2012 | Gantt Chart | sbs1 |
| V1.3 | N/A | 26/10/2012 | Use Case Diagrams | cew10 |
| V1.4 | N/A | 1/11/2012 | Risk Analysis | cab27 |
| V1.5 | N/A | 1/11/2012 | Overview and introduction | sbs1 |
| V1.6 | N/A | 19/01/2013 | Added Use-Case diagram description  Proofread and corrected mistakes. | cew10 |

# Testing Document

## 1. INTRODUCTION

### 1.1 Purpose of this Document

The purpose of this document is to provide a detailed description of the testing to be carried out to ensure the production of an efficient and effective application. It should be read in the context of the Group Project 2012 taking into account the details of the Group Project assignment.

### 1.2 Scope

This document will present the testing required to complete the system. It will also cover the procedure for reporting test results. It will not include the procedure for fixing failed tests. The module testing involved in this project will be done using unit testing as described in the test plan.

### 1.3 Objectives

The objectives of this document are:

* To provide a detailed test specification that can be referred to throughout testing of the system.
* To describe the test result reporting procedure.
* To declare the format of testing and the information which must be supplied in test specifications and reports.

## 2. TEST SPECIFICATION

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Ref | Req being tested | Test Content | Input | Output | Pass Criteria |
|  | FR1 | Check that a user can log in with a valid account | Valid username and password | The log in should be successful | The user can log in |
|  | FR1 | Check that a user cannot log in using an invalid account | Invalid username and password | The log in should be unsuccessful | The user is unable to log in |
|  | FR1 | Check that a user can register an account via a browser | Acceptable username and password | The account should be registered successfully | The users account has been created |
|  | FR1 | Check that a user can only register with an appropriate username and password | The username will have to consist of only alphabetical letters and numerical numbers. Anything else would be considered invalid for example adding punctuation marks such as “#”. In regards to the password anything can be inputted as it will be valid.  Both username and password will have to be a minimum of 6 characters. | The account should not be created and an error message should appear stating that the inputted data is invalid. | The account is not created |
|  | FR1 | Check that both username and passwords fields must be filled | One or both Fields are left blank | The login should be unsuccessful and the user should be prompted to fill all fields | The login is unsuccessful and the user is prompted |
|  | FR2 + FR9 | Check that a friend request can be sent and received | A friend request will be sent to another user | The request should appear on the other user's account | The request appears correctly |
|  | FR2 + FR9 | Check that a friend request can be accepted | Select the accept option on the received friend request | The users should appear on each others friends list and a confirmation message sent | The users appear on each others friends list and receive a confirmation message |
|  | FR2 + FR9 | Check that a friend request can be declined | Select the decline option | A message should appear saying the request has been declined | The users do not appear on each others friends lists and a declination message is sent |
|  | FR3 | Check that monster list is displayed correctly | The monster page will be displayed | All monsters owned by that user should be displayed | The monsters owned by that user should be displayed |
|  | FR3 | Check that monster attributes are displayed correctly | The monster page will be displayed | The attributes of each monster on the list should be displayed correctly | The monster attributes are displayed correctly |
|  | FR3 | Check that a new user is allocated a monster and an amount of virtual currency | Create a new user account | The user should be allocated a monster and a sum of virtual currency | The user is allocated a monster and the correct amount of virtual currency |
|  | FR3 | Check that the monster allocation is random | Create multiple user accounts | The basic monster allocation should not be the same for each user | The monster allocation is random |
|  | FR4 | Check that the system allocates a prize for a challenge appropriate to monster | Create a new monster and ensure that the prize is low relative to more powerful monsters | The prize for winning a challenge should scale up relative to monster level | The prize must scale relative to monster level |
|  | FR4 | Check the system puts the two correct monsters together for a battle based on user input | The user selects one of their own monsters and one of their opponents monsters to battle | The two selected monsters are put together to battle | The two correct monsters are selected to battle |
|  | FR4 | Check that the user can accept a battle | Once challenged to a battle the user selects accept | The battle should go ahead | The battle goes ahead |
|  | FR4 | Check that the user can decline a battle | Once challenged to a battle the user selects decline | The battle should be cancelled | The battle is cancelled and does not go ahead |
|  | FR5 | Check that users from different servers can add each other as friends | A user will attempt to add a user from a different server to their friends list | The request should appear correctly on the second users requests page | The request displays correctly |
|  | FR5 | Check that users from different servers can accept requests | A user will attempt to add a user from a different server to their friends list | After accept is selected the users will appear on each others friends lists | The users appear on each others friends lists |
|  | FR5 | Check that users from different servers can decline requests | A user will attempt to add a user from a different server to their friends list | The request should be declined | The request is declined and the users do not appear on each others friends lists |
|  | FR5 | Check that users from different servers can buy monsters | One user will attempt to buy a monster from the other server | The transaction should succeed and one user should gain the monster and one gain the value in virtual currency | The transaction succeeds and the monster and the currency are exchanged |
|  | FR5 | Check that users from different servers can arrange a monster breeding | One user will send a breeding request to a user on a different server | The breeding should go ahead | The breeding goes ahead |
|  | FR5 | Check that users from different servers can arrange a fight | One user will send a fight request to a user on a different server | The fight should happen correctly (winner chosen and prizes allocated correctly) | The winner and prize money are allocated correctly |
|  | FR6 | Check that the user can buy a monster | User selects the buy option on a monster for sale | The user should gain the monster and lost the currency value of the monster | The user gains the monster and loses the value of the monster |
|  | FR6 | Check that the user can sell a monster | The user puts a monster up for sale | The monster is correctly displayed as for sale and once the sale takes place the monster is gone and the user has gained the value of the monster | The monster is put up for sale and sold successfully |
|  | FR6 | Check that the user can remove a friend | On the friends list select the remove friend option from a friend | The friend in question should be removed from the friends list and the user removed from theirs | The friend is removed successfully |
|  | FR6 | Check that the user can offer a monster for breeding | The user will select the offer for breeding option on a monster | The monster will display as up for breeding correctly | The monster displays as up for breeding |
|  | FR6 | Check that the user can purchase breeding with a monster | The user will select the breed option on a monster available to breed | The breeding should go ahead and the cost deducted from the users bank | The breed goes ahead properly and the user is appropriately charged |
|  | FR6 | Check that the user can unregister | The user will select the unregister option | The users account should be unregistered and no longer be valid | The users account is no longer valid |
|  | FR7 | Check that on startup the options to login and register are displayed | The user will start up the program in their browser | The options to log in or register should be displayed | The options to log in and register are displayed |
|  | FR7 | Check that once logged in the user can log out | Once logged in the user will select the log out option | The user should be logged out of the system | The user is logged out of the system |
|  | FR8 | Check that once logged in the user has a list of their monsters statuses | The user will log in and go to their monsters page | The user should see a list of their monsters (with health, diseases etc) | The user is presented with a complete list of their monsters plus their statuses |
|  | FR8 | Check that once logged in the user has a list of their friends | The user will log in and go to their friends page | The user should see a list of their friends | The user sees a correct list of their friends |
|  | FR8 | Check that once logged in the user has a list of monsters available for breeding | The user will log in and go to their friends page | The user should see a list of monsters that are available for breeding | The user sees a correct list of monsters available for breeding |
|  | FR8 | Check that once logged in the user has a list of monsters for sale | The user will log in and go to their friends page | The user should see a list monsters that are available to be bought | The user sees a correct list of monsters available for sale |
|  | FR8 | Check that once logged in the user can see a list of all their requests (breeding, challenges etc) with prize money if applicable | The user will log in and go to their friends page | The user should see a list of their requests plus prizes | The user sees a full list of their requests plus the prizes available |
|  | FR10 | Check that after a fight the users competitors monsters list is updated | Once logged in the user will take part in a battle | The monster list of all competitors should be updated | The competitors monster list is updated (losers removed from the list) |
|  | FR10 | Check that after a battle has been won the prize money is transferred to the users account | Once logged in the user will take part in a battle and win | The prize should then be in the users account and their monsters updated to account for injuries | The prize money is in the users account and their monsters updated for injuries |
|  | FR11 | Check that the user can see the wealth of all friends | Once logged in the user navigates to the friends page | All friends wealth should be displayed in order of wealth | Friends are displayed in correct order of wealth |

## 3. REFERENCES

*[1] QA Document SE.QA.01 - Quality Assurance Plan.*

*[2] QA Document SE.QA.03 – General Documentation Standards.*

*[3] QA Document SE.QA.06 – Test Procedure Standards.*

*[4] QA Document SE.QA.08 – Operating Procedures and Configuration Management Standards.*

## 4. DOCUMENT CHANGE HISTORY

| Version | CCF No. | Date | Changes Made to Document | Changed By |
| --- | --- | --- | --- | --- |
| 1 | N/A | 14/11/12 | Draft | C.Bird |
| 2 | N/A | 29/01/13 | Draft | I.Mungul |

# Design Document

## Introduction

### Purpose of this Document

The purpose of this document was to provide an insight as to how we proceeded in creating the diagrams required for the group project we had been assigned to do. This document will present to you all the information, from descriptions to designs, which “Group 15” had created in regards to the design aspect of this project.

### Scope

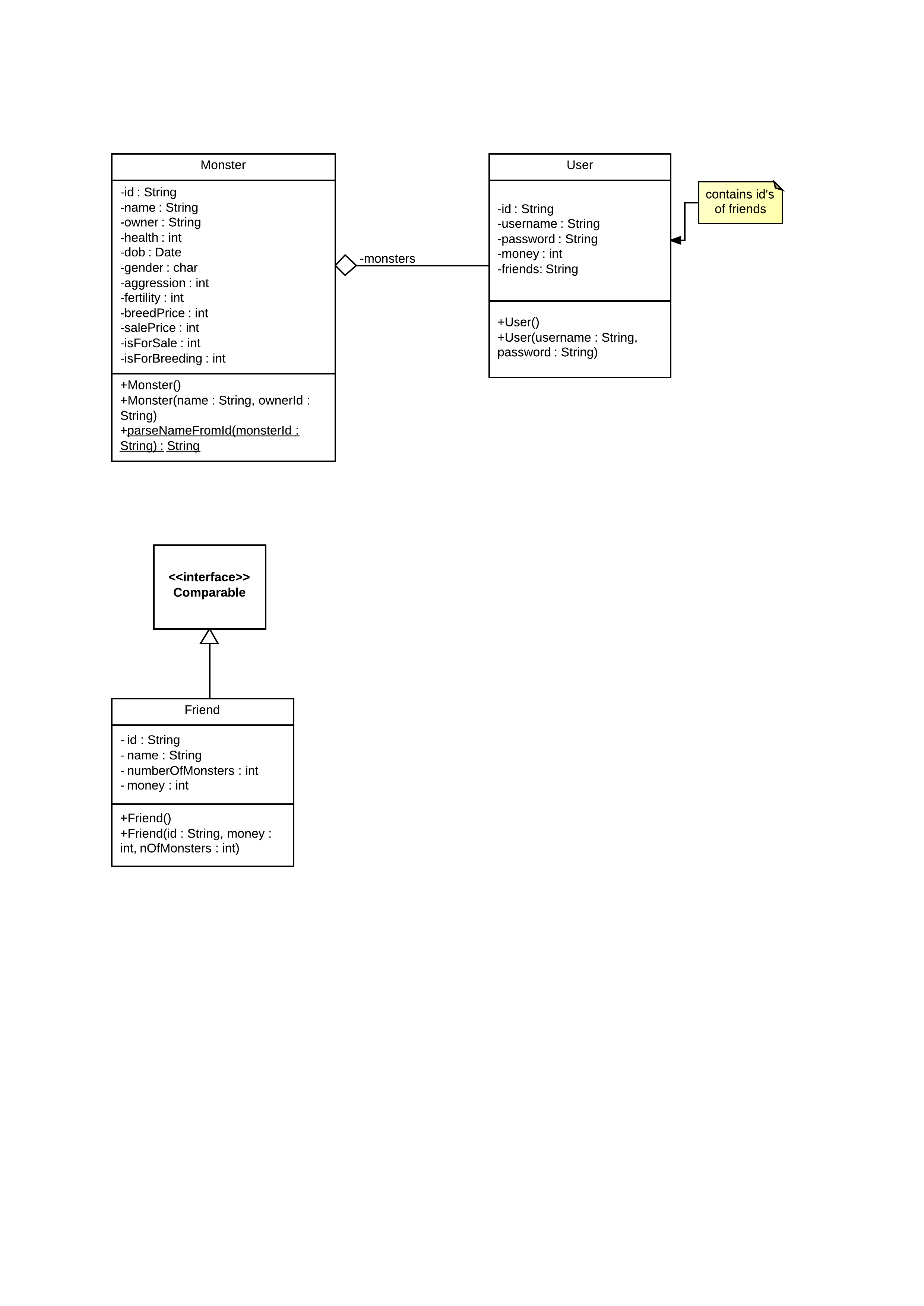
This document shows all the diagrams, methods and descriptions regarding the design factor of this project. It reveals how the diagrams all inter-link to each other from the server side to the design. The descriptions of the classes used are clearly explained in this document, as well as the methods we had used, to give you an understanding of how we progressed in this assignment.

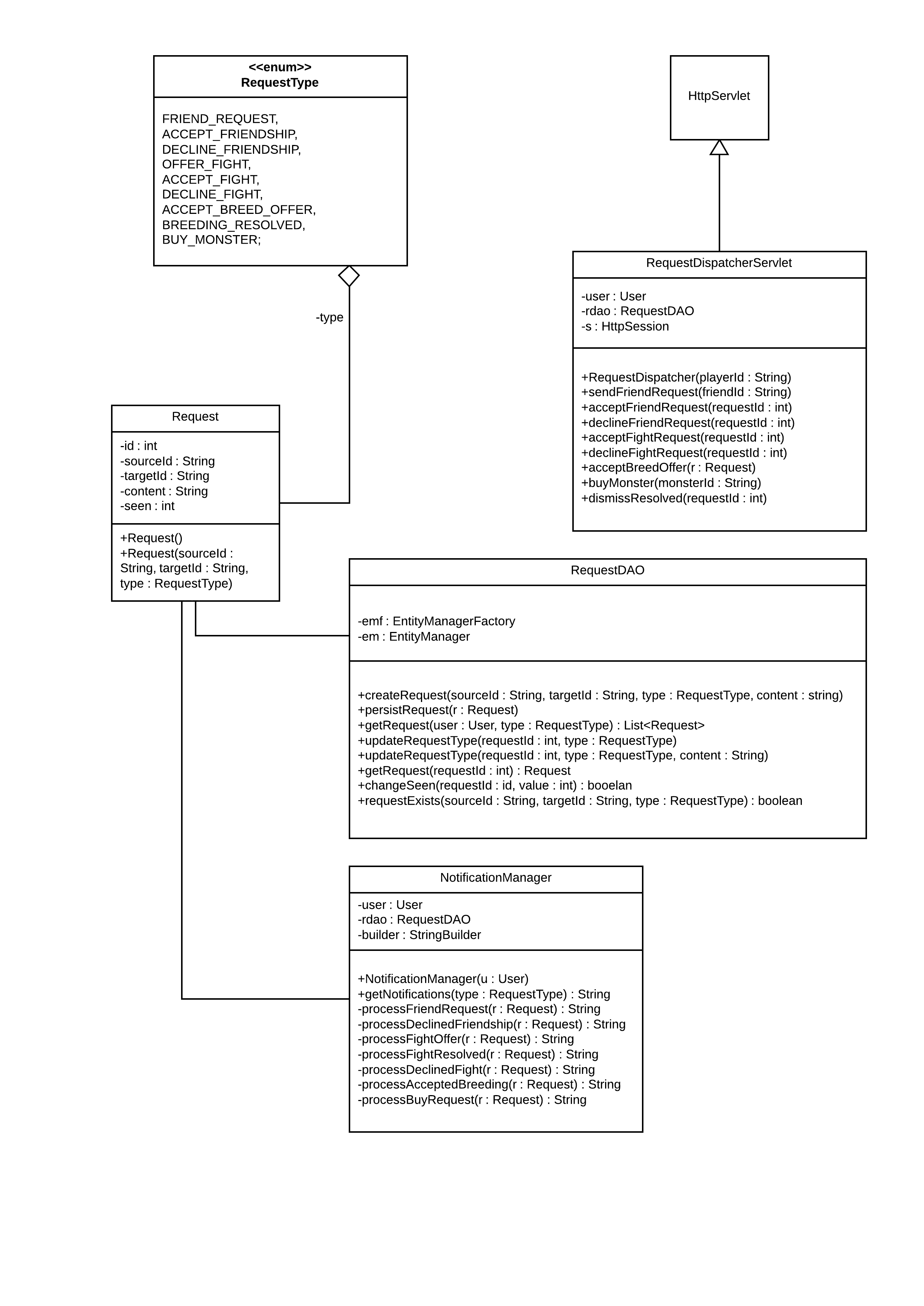
### Objectives

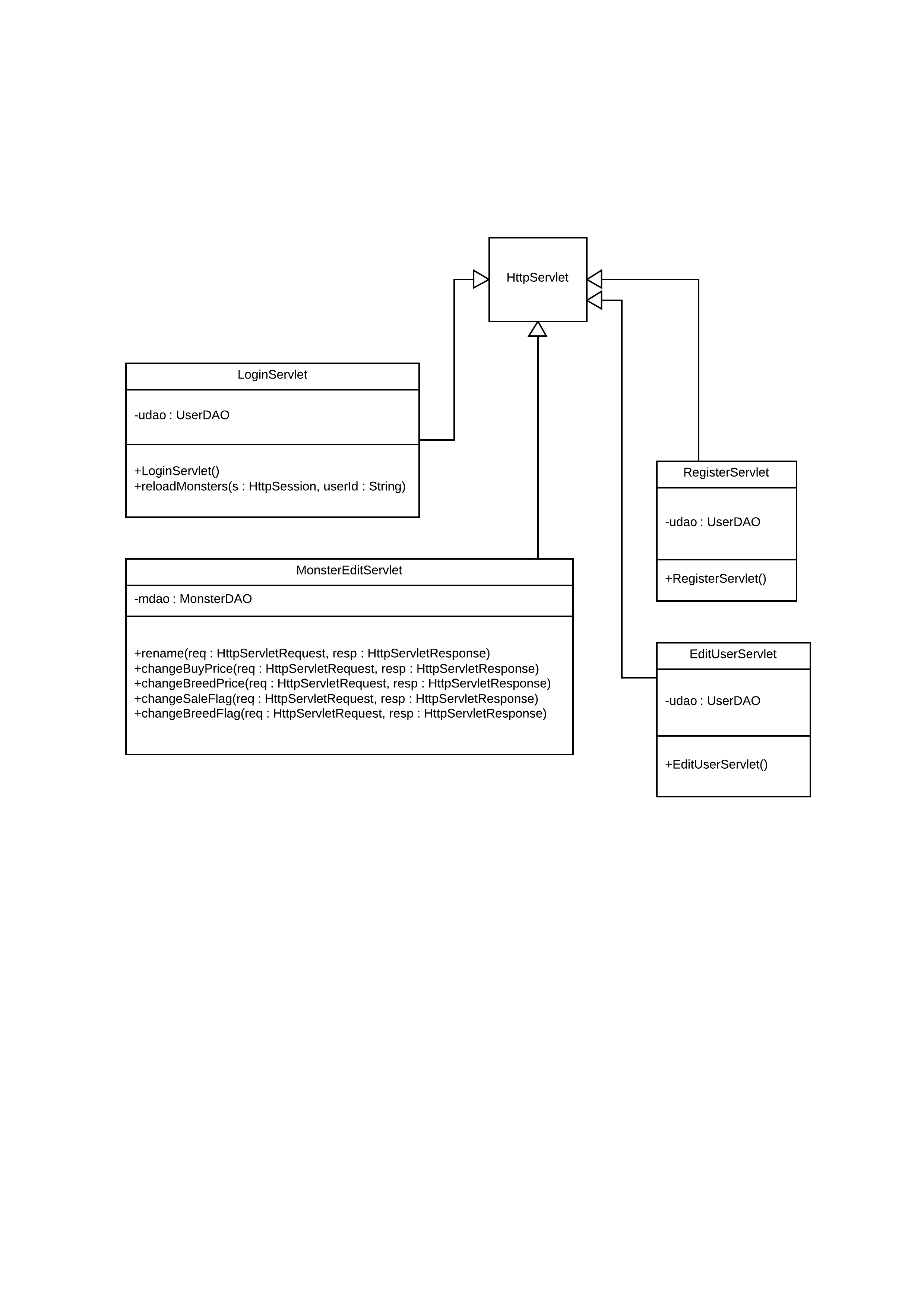
The priority objective for the design team was to create the necessary resources for the application to the point that it worked efficiently, as well as having an impressive appearance. The software structure, components, interfaces and the data revolving around the design factors would all have to be drawn up and created by the team ensuring it meets the requirements of the group as a whole.

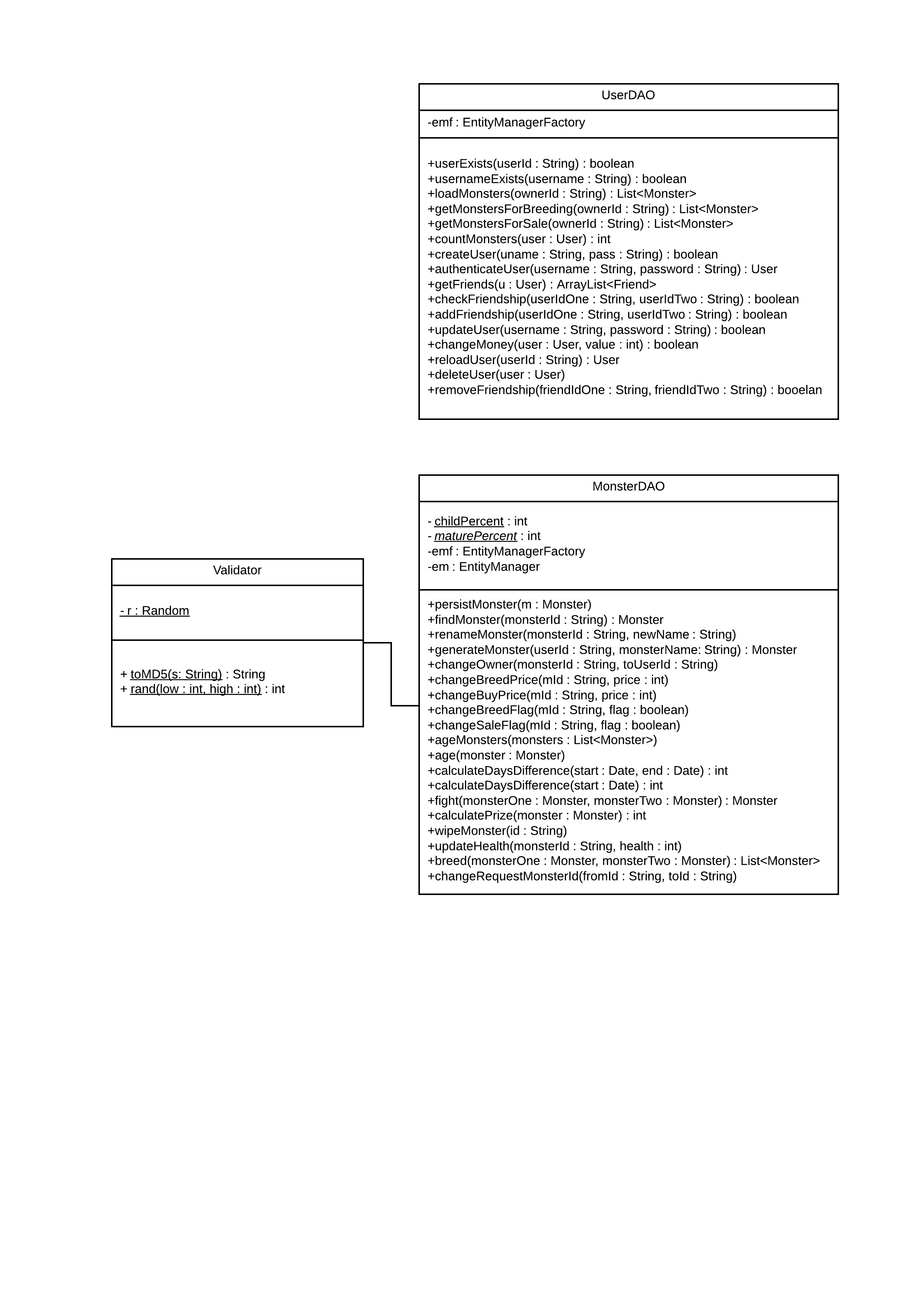
## Decomposition Description

### Class Diagram









### Class Description

**Monster** – This class contains the attributes of the monster and methods to set monsters for breeding or for sale.

**User** – This class contains the details of the user, including username and password required to log them in. It also contains variables for the amount of money they have and their list of friends. It contains methods to set the password, get money (when account is created, by winning fights, selling, buying and breeding monsters), adding friends and deleting friends, adding monsters (by buying or breeding) and deleting monsters (when they lose fights or die of old age).

**UserDAO/User** – These classes manages the user accounts. It contains methods to create an account, remove an account, reset an account and authenticate the user and contains methods to obtain data from the server

**MonsterDAO/Monster** – These classes manages all methods when fetching and manipulating data from the database. It has certain functions, which are vital to the monsters fighting statistics, such as *age()*, but then also has certain methods for fighting and breeding monsters.

**RequestDAO/Request** – These classes manage all data manipulation when it comes to finding, creating or deleting requests from the database, and assigning them to different Users/Monsters

**RequestDispatcherServlet** – handles all input for setting up requests for monsters to fight/breed or be sold, while also serving as a point for users to add friends. It has been designed in such a manor that a simple layer on top to parse JSON is only needed for manipulating incoming/outgoing data.

### Mapping from requirements to Classes

|  |  |
| --- | --- |
| **Requirement** | **Classes providing requirement** |
| FR1 | AccountManager |
| FR2 | DataManager, Friend |
| FR3 | User, DataManager |
| FR4 | RequestDispatcher, FightRequest, FightResolver |
| FR5 | RequestDispatcher, Request |
| FR6 | AccountManager, User, Monster |
| FR7 | AccountManager, User, index.jsp |
| FR8 | JSP pages |
| FR9 | RequestDispatcher, friends.jsp |
| FR10 | FightResolver, FightRequest, fight.jsp |
| FR11 | DataManager |
| EIR1, PR1, PR2 | JSP Pages |

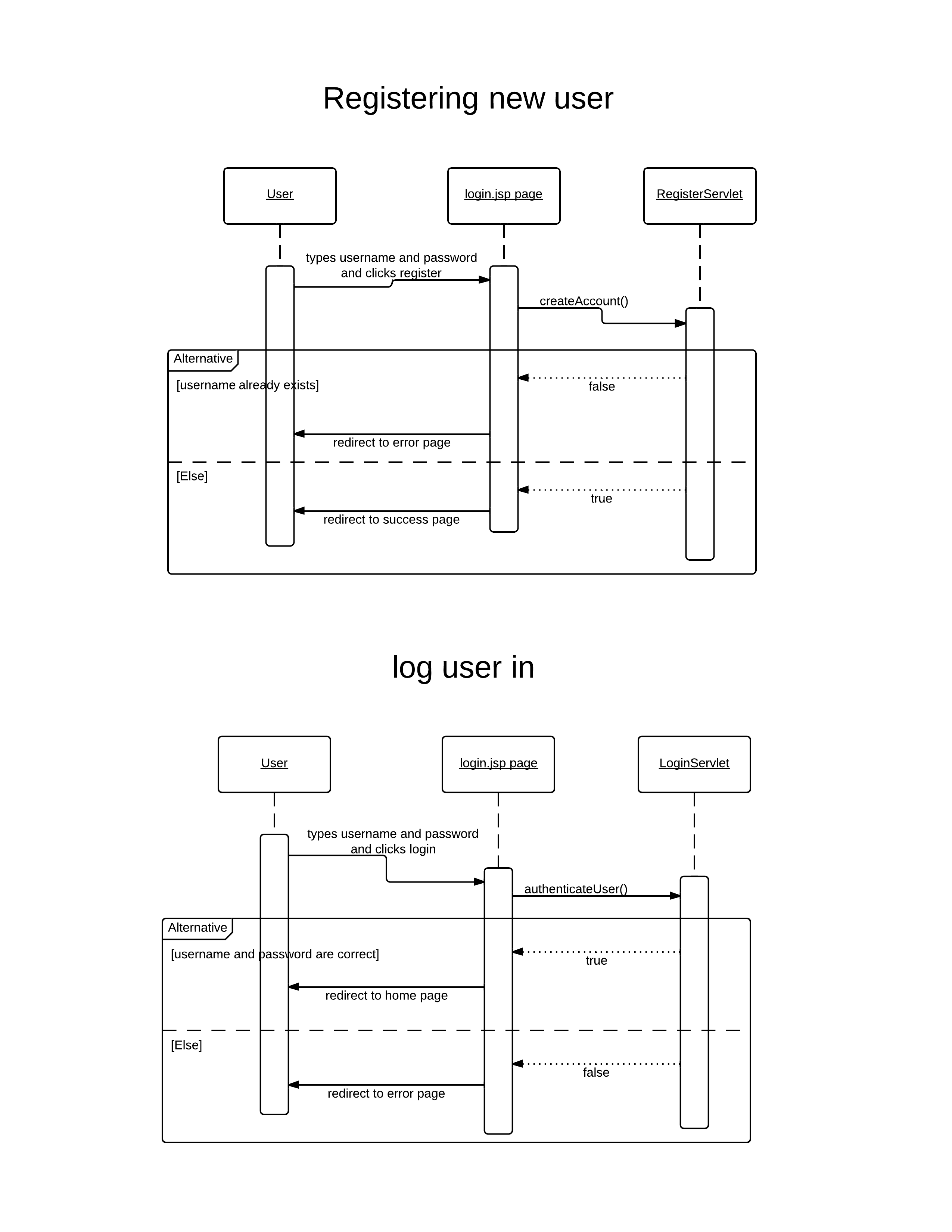
## 

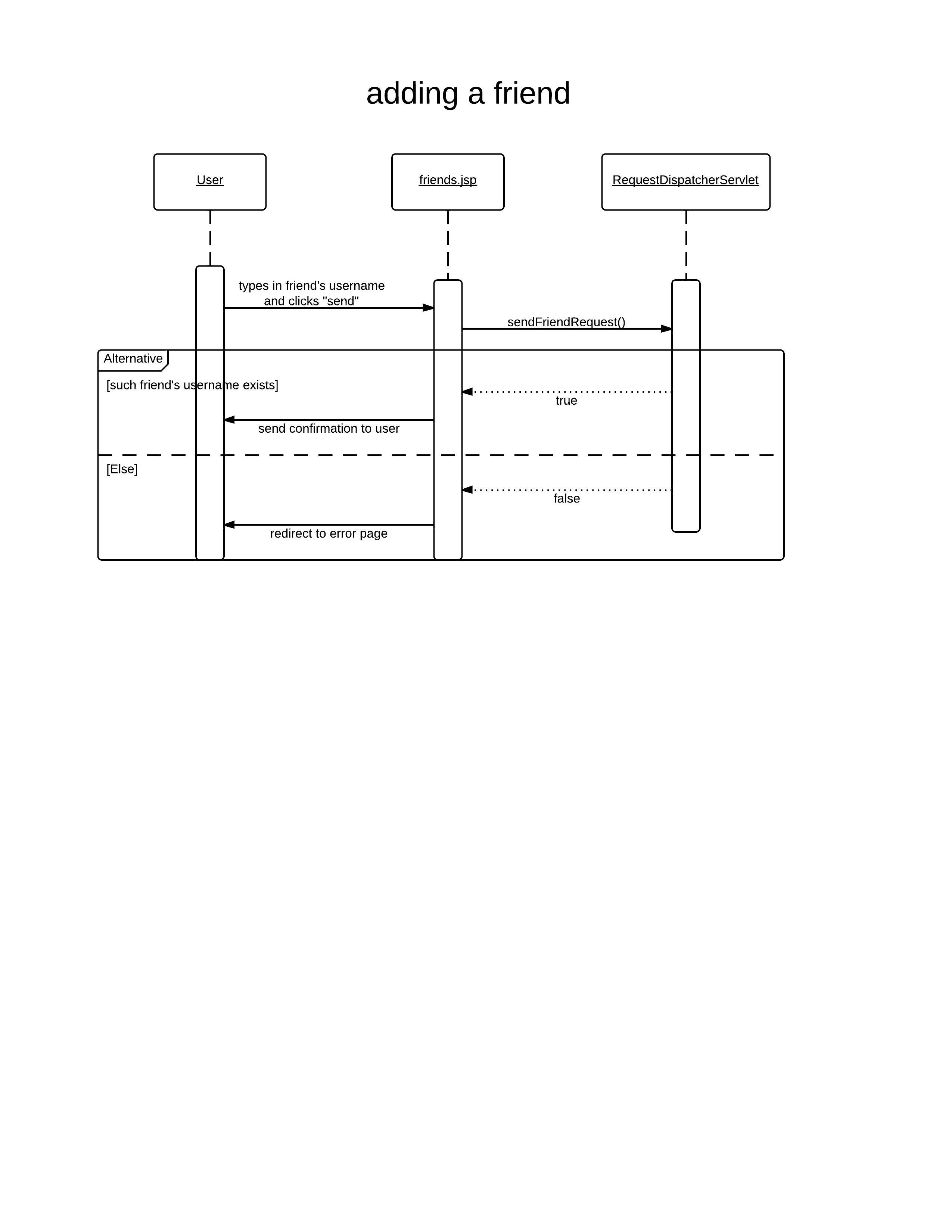
## Dependency Description

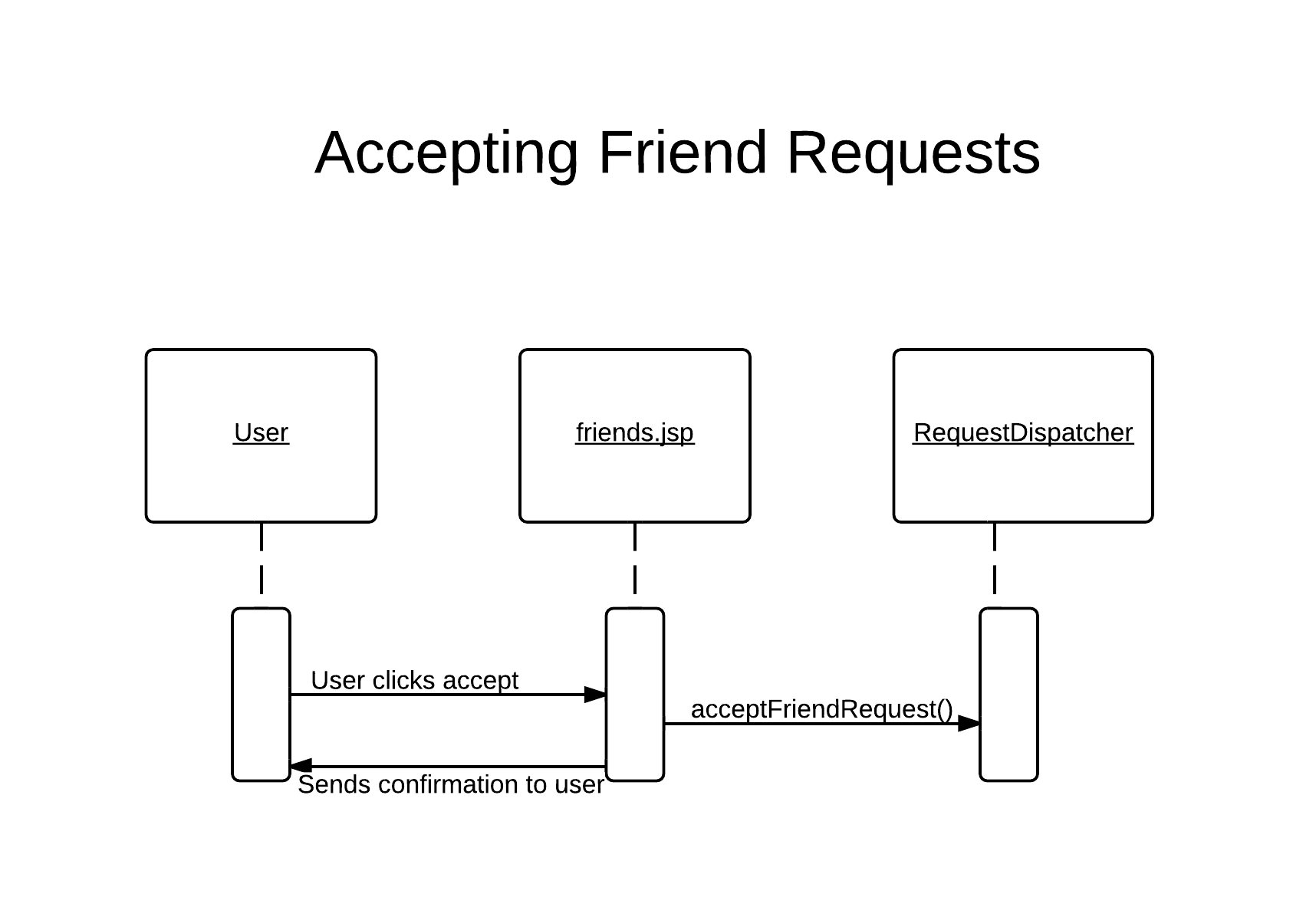
### Component Diagrams

## Detailed Design

### Sequence Diagrams









# Macintosh HD:Users:user:Downloads:BasicSequenceDiagram (2).png

# Macintosh HD:Users:user:Downloads:BasicSequenceDiagram (3).png

# Macintosh HD:Users:user:Downloads:BasicSequenceDiagram (4).pngChange log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF Number | Date | What Changed | User |
| 1.0 | N/a | 02.12.2012 | First Draft | imm5 |
| 1.1 | N/a | 07.12.2012 | Brought up to standard. | cew10 |
| 1.2 | N/a | 30.01.2013 | Proofread and corrections | cew10 |
| 1.3 | N/A | 14/02/2013 | Updated with new design | Sbs1 |

# Maintenance Document

## Project Description

This program is an online game to play for children and adolescents can play, where they own a “Monster Farm”, and are able to Fight, Breed and Sell these monsters to other players on the server. It is all written in Java, utilising certain modules of J2EE, such as JPA (Java Persistence Architecture) and Glassfish server, and the website is majorly written in xHTML Transitional, CSS2 and Javascript.

## Program Structure

The program was developed with a strong MVC. Therefore:

* All data models of the database, including JPA references, should be placed in the model package
* All Servlets should be placed in the view package. Please note that JSPs don’t live in this file, and only reside in the WEBCONTENT directory.
* All controllers, including Data Access Objects and miscalleneous functions should be placed here.

### Method calling

The start of a method call always starts from a Servlet. For example, all data sent from the login form gets sent to the LoginServlet. From there, it will check all values using a DAO against the related model. If everything goes well it, the user is sent to the target page, otherwise an error will display to the user.  
  
For more details on the overall design, please view the Design Document.

## Algorithms

There are three main algorithms which are important to the game: Aging, Fighting and Breeding.

### Aging

For aging the monster, we have designed the solution to work out the age from base values, which are found in the database; from there it will age day by day. We also have age ranges for all monsters: child (0-4), adult(5-11) and elderly(12+); during these phases of the monsters life, different stats will increase or decrease depending on how old it is. The values are as follows:

* Children have a 5% increase to all stats (not including health)
* Adults have a 2% increase to Strength and Aggression; Fertility and Health stay the same
* The elderly monsters stats stay the same, but the health of the monster decreases at 15 - (strength + aggression + fertility7) points each day, until it hits 0

### Fighting

The main fighting algorithm has three parts to it: Attack, Health and Luck, which all combine to give a normalised value between which monster will win. The formula for calculating the value is:  
hit =Strength  Aggression1000 + health20 + rand(1, 10). This is classed as a 1 hit system for the monsters, and the winning monster will have his health decreased by a percentage based on the hit which was given by the other monster.

### Breeding

For breeding monsters, we decided that we will follow a simplified version of reproduction in the real world. Instead of having dominant and recessive genes which might pop out at a certain point, we created that there will be an equal chance that child will get the weaker parents stat to getting the stronger parents stat. We also gave an element of randomness for mutations, which will choose and random number between 20 and 70 and not at all biased on the parents statistics  
  
For more details on the overall design, please view the Design Document.

## Data Structures

All data structures in this solution (apart from Friend.java) are based on MySQL table entities - and vice versa, so using JPA mapping between the three tables are incredibly easy, and as the structure is written in Java, moving the table structure between databases/servers is simple  
  
For more details on the overall design, please view the Design Document.

## Issues and Improvements

All issues and improvements to the software can be found at the Github issue tracker. This is where all maintainers and developers will post up bug issues and new enhancements, which will then be handled by individual maintainers.

## Making Changes

When making changes to the codebase, make sure you have forked the repository to your own account on Github, this will make merging requests easier when merging your additional code to the master branch of the main repo. Also, when you are fixing an issue, please assign yourself to the issue!

## Physical Limitations

There are several physical limitations of the software at present. One of which is the database server MySQL. If the site does take a large amount of users in a short amount of time, MySQL is not the most efficient server for such a task. However using JPA shouldn’t be a problem with upgrading the database to a NoSQL alternative.

## Rebuilding and Testing

To rebuild, I suggest using a J2EE environment, which allows for automatic compiling and deploying of the .war file to the glassfish server, such as Netbeans or Eclipse with certain addons. The server requires that you have installed the J-Connector for MySQL into the lib folder of the glassfish server, and that you’ve created a pool of connections and a driver called ‘jdbc/mysql’ which has access to the database. (NB: the name of the driver can change if you edit the persistance.xml file).

Due to issues with connection pools and Glassfish, JUnit tests **cannot** run on data, which has been fetched from the database. This is a known issue and will be fixed in the near future.

# Completed Testing Document

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Ref | Req being tested | Test Content | Input | Output | Pass Criteria | Pass/Fail |
|  | FR1 | Check that a user can log in with a valid account | Valid username and password | The log in should be successful | The user can log in | Pass |
|  | FR1 | Check that a user cannot log in using an invalid account | Invalid username and password | The log in should be unsuccessful | The user is unable to log in | Pass |
|  | FR1 | Check that a user can register an account via a browser | Acceptable username and password | The account should be registered successfully | The users account has been created | Pass |
|  | FR1 | Check that a user can only register with an appropriate username and password | The username will have to consist of only alphabetical letters and numerical numbers. Anything else would be considered invalid for example adding punctuation marks such as “#”. In regards to the password anything can be inputted as it will be valid.  Both username and password will have to be a minimum of 6 characters. | The account should not be created and an error message should appear stating that the inputted data is invalid. | The account is not created | Pass |
|  | FR1 | Check that both username and passwords fields must be filled | One or both Fields are left blank | The login should be unsuccessful and the user should be prompted to fill all fields | The login is unsuccessful and the user is prompted | Pass |
|  | FR2 + FR9 | Check that a friend request can be sent and received | A friend request will be sent to another user | The request should appear on the other user's account | The request appears correctly | Pass |
|  | FR2 + FR9 | Check that a friend request can be accepted | Select the accept option on the received friend request | The users should appear on each others friends list and a confirmation message sent | The users appear on each others friends list and receive a confirmation message | Pass |
|  | FR2 + FR9 | Check that a friend request can be declined | Select the decline option | A message should appear saying the request has been declined | The users do not appear on each others friends lists and a declination message is sent | Pass |
|  | FR3 | Check that monster list is displayed correctly | The monster page will be displayed | All monsters owned by that user should be displayed | The monsters owned by that user should be displayed | Pass |
|  | FR3 | Check that monster attributes are displayed correctly | The monster page will be displayed | The attributes of each monster on the list should be displayed correctly | The monster attributes are displayed correctly | Pass |
|  | FR3 | Check that a new user is allocated a monster and an amount of virtual currency | Create a new user account | The user should be allocated a monster and a sum of virtual currency | The user is allocated a monster and the correct amount of virtual currency | Pass |
|  | FR3 | Check that the monster allocation is random | Create multiple user accounts | The basic monster allocation should not be the same for each user | The monster allocation is random | Pass |
|  | FR4 | Check that the system allocates a prize for a challenge appropriate to monster | Create a new monster and ensure that the prize is low relative to more powerful monsters | The prize for winning a challenge should scale up relative to monster level | The prize must scale relative to monster level | Pass |
|  | FR4 | Check the system puts the two correct monsters together for a battle based on user input | The user selects one of their own monsters and one of their opponents monsters to battle | The two selected monsters are put together to battle | The two correct monsters are selected to battle | Pass |
|  | FR4 | Check that the user can accept a battle | Once challenged to a battle the user selects accept | The battle should go ahead | The battle goes ahead | Pass |
|  | FR4 | Check that the user can decline a battle | Once challenged to a battle the user selects decline | The battle should be cancelled | The battle is cancelled and does not go ahead | Pass |
| FR5 - 1 | FR5 | Check that users from different servers can add each other as friends | A user will attempt to add a user from a different server to their friends list | The request should appear correctly on the second users requests page | The request displays correctly | Fail (Server-server communication does not work) |
|  | FR5 | Check that users from different servers can accept requests | A user will attempt to add a user from a different server to their friends list | After accept is selected the users will appear on each others friends lists | The users appear on each others friends lists | Fail (See explanation for test FR5 -1) |
|  | FR5 | Check that users from different servers can decline requests | A user will attempt to add a user from a different server to their friends list | The request should be declined | The request is declined and the users do not appear on each others friends lists | Fail (See explanation for test FR5 -1) |
|  | FR5 | Check that users from different servers can buy monsters | One user will attempt to buy a monster from the other server | The transaction should succeed and one user should gain the monster and one gain the value in virtual currency | The transaction succeeds and the monster and the currency are exchanged | Fail (See explanation for test FR5 -1) |
|  | FR5 | Check that users from different servers can arrange a monster breeding | One user will send a breeding request to a user on a different server | The breeding should go ahead | The breeding goes ahead | Fail (See explanation for test FR5 -1) |
|  | FR5 | Check that users from different servers can arrange a fight | One user will send a fight request to a user on a different server | The fight should happen correctly (winner chosen and prizes allocated correctly) | The winner and prize money are allocated correctly | Fail (See explanation for test FR5 -1) |
|  | FR6 | Check that the user can buy a monster | User selects the buy option on a monster for sale | The user should gain the monster and lost the currency value of the monster | The user gains the monster and loses the value of the monster | Pass |
|  | FR6 | Check that the user can sell a monster | The user puts a monster up for sale | The monster is correctly displayed as for sale and once the sale takes place the monster is gone and the user has gained the value of the monster | The monster is put up for sale and sold successfully | Pass |
|  | FR6 | Check that the user can remove a friend | On the friends list select the remove friend option from a friend | The friend in question should be removed from the friends list and the user removed from theirs | The friend is removed successfully | Pass |
|  | FR6 | Check that the user can offer a monster for breeding | The user will select the offer for breeding option on a monster | The monster will display as up for breeding correctly | The monster displays as up for breeding | Pass |
|  | FR6 | Check that the user can purchase breeding with a monster | The user will select the breed option on a monster available to breed | The breeding should go ahead and the cost deducted from the users bank | The breed goes ahead properly and the user is appropriately charged | Pass |
|  | FR6 | Check that the user can unregister | The user will select the unregister option | The users account should be unregistered and no longer be valid | The users account is no longer valid | Pass |
|  | FR7 | Check that on startup the options to login and register are displayed | The user will start up the program in their browser | The options to log in or register should be displayed | The options to log in and register are displayed | Pass |
|  | FR7 | Check that once logged in the user can log out | Once logged in the user will select the log out option | The user should be logged out of the system | The user is logged out of the system | Pass |
|  | FR8 | Check that once logged in the user has a list of their monsters statuses | The user will log in and go to their monsters page | The user should see a list of their monsters (with health, diseases etc) | The user is presented with a complete list of their monsters plus their statuses | Pass |
|  | FR8 | Check that once logged in the user has a list of their friends | The user will log in and go to their friends page | The user should see a list of their friends | The user sees a correct list of their friends | Pass |
|  | FR8 | Check that once logged in the user has a list of monsters available for breeding | The user will log in and go to their friends page | The user should see a list of monsters that are available for breeding | The user sees a correct list of monsters available for breeding | Pass |
|  | FR8 | Check that once logged in the user has a list of monsters for sale | The user will log in and go to their friends page | The user should see a list monsters that are available to be bought | The user sees a correct list of monsters available for sale | Pass |
|  | FR8 | Check that once logged in the user can see a list of all their requests (breeding, challenges etc) with prize money if applicable | The user will log in and go to their friends page | The user should see a list of their requests plus prizes | The user sees a full list of their requests plus the prizes available | Pass |
|  | FR10 | Check that after a fight the users competitors monsters list is updated | Once logged in the user will take part in a battle | The monster list of all competitors should be updated | The competitors monster list is updated (losers removed from the list) | Pass |
|  | FR10 | Check that after a battle has been won the prize money is transferred to the users account | Once logged in the user will take part in a battle and win | The prize should then be in the users account and their monsters updated to account for injuries | The prize money is in the users account and their monsters updated for injuries | Pass |
|  | FR11 | Check that the user can see the wealth of all friends | Once logged in the user navigates to the friends page | All friends wealth should be displayed in order of wealth | Friends are displayed in correct order of wealth | Pass |

# Final Report

## Management Summary

The project overall was a success, especially with the final solution we created; however, there were certain problems which we encountered during the time on development on the solution.  
  
The state of the project currently is that all features (bar dying at the end of ageing) are implemented on a single server setup, with minimal amount of bugs across the system. However, we have not implemented Server-to-Server integration on our solution, but there is a framework with a single entry point where a remote server can access data. We decided that dropping this functionality would be better for the overall project, as having a rock solid platform for the functionality to be integrated at a later point is far more important than a lackluster approach to all the programming and have a half working solution by the submission date.  
  
I feel we did work well in a team, with everyone knowing what needed to be done, and after a brief lesson on how Github and its issue tracker worked, it gave a good pace to the development. Also we managed to split into a “paired programming” methodology in Integration and Testing week, which worked wonders with the productivity of the overall members which might have been lacking at the start of the week. However, all this learning was at quite a late point in the project, and if we learnt these invaluable lessons at an earlier point, we would have had a better working project at the end of the I&T week.  
  
It was a shame that Charlie Bird did leave us at a point, as he did give contribute to the project at points, and due to his unforeseen departure, I did spend a couple of hours understanding what he put into the project with no handover message from him.  
  
Apart from these minor issues and problems we encountered during our time on the project, I feel that we have created a piece of software that we should be proud of!

## Timeline of the Project

### Requirements Document

* Job allocation was made for researching different types of software we needed to utilise for creating the solution, such as IDEs, deployment solutions and version control systems to name a few. We used
  + Eclipse for the development
  + Glassfish as the deployment solution
  + MySQL for the storage of data
  + JPA for accessing the data
  + Git for Version Control
  + Microsoft Word for documentation
* Initial job allocation for members of the group, which gave a basis for the rest of the project.
* Mock up of the interface was created for the requirements specification, which went on to inspire the current theme we used in the final solution

### Testing Document

* Grant was allocated to design a testing scheme for integration and testing week, and he and Charlie wrote the testing document according to the functional requirements specified by the client.
* During this time, Kamil rewrote the UI to implement a more stylish and finished look onto the UI ready to be implemented.

### Design Document

* Fiona and Imran were tasked with the design document, with Kamil often lending a hand with explaining and quality assurance on the overall document.
* Sam had been working on implementation of authentication and registration of users ready for the display.

### Integration and Testing Week

#### Monday

* First meeting at 9AM with a plan of action for the rest of the week
* Morning was used for setting up development and production servers and sorting out arising issues
* Afternoon we implemented retrieving monsters and friends from the database when a user has logged in

#### Tuesday

* Morning was used for testing the first milestone with registration and login for the testing team while the development team carried on implementing the friendship options, including adding and deleting friendships
* Afternoon, I got the documentation team to start responding to feedback on the documentation we have already submitted, while giving an outline for the final submission documents.

#### Wednesday

* Morning, the testing team tried out the new bug fixed login system with the added functionality for friendships, During this time, Kamil, Sam and Andy were working with getting the RequestDispatcher framework working with Fighting and Breeding.
* Afternoon, we started implementing the main monster algorithms including breeding and fighting, which we perfected when working later into the night.

#### Thursday

* Most features were implemented by the morning of the Thursday, so the testing team worked on bug testing the system at a whole, and started submitting bug issues on the Github web interface. During this time, Andy was spending time working with the graphics and layout, while Kamil and Sam were working on bug fixes and last bits of the required functionality.
* In the afternoon, all submitted documentation was assessed and brought upto a final draft.

#### Friday

* The morning was used to clean up some final bugs before submitting all code to blackboard and then running through the acceptance tests in the afternoon.

## Final Report

* Kamil and Sam rewrote parts of the UML design which was flawed when implementing the original design document.
* Sam help finish and finalise the documents ready to be submitted.

## Final State of the Project

The final state of the project is one of near completeness when running with just one server. The only issues we have found is minor bug problems (such as parsing user IDs to the jsp to display not working correctly) and an issue with the way we implemented the

## Performance of Members

### Sam Sherar <sbs1> - Team Leader

I feel that I worked well managing the team during the time across the project with the limited tools I had, and I am happy with the amount of work I submitted during integration and testing week towards the main codebase. However I could have been more available during the first semester for more questions from the team.

### Kamil Mrowiec <kam20> - Co-Team Leader

I feel that Kamil worked incredibly well over the time spent on the project. He always very prompt to all meetings and had always had something to contribute to the project. He also was not afraid to challenge certain decisions, which spawned some great ideas!

### Fiona Joseph Samy <fij1> - QA Manager

Fiona is a hard worker, especially when it comes to researching new material and other ideas. I felt that she could have been more productive if she wasn’t afraid to ask me questions about certain topics, but overall I am happy with the amount of work she produced.

#### Andy Watkins <ajw14> - Design Team

I feel that Andy did contribute to the project by creating most of the User Interface from design to production, but he did not produce as much as I was hoping for, as certain social networking sites easily distracted him. However, the work he did produce was solid and adhering to the coding standards we implemented.

### Imran Mungul <Imm5> - Design Team

Imran was a hard worker to start, but I felt his motivation dipped over the course of the project, and by Integration and Testing week, it was hard work to get him going on a task. But after he got going, the quality of work and attention to detail was superb, and trying to understand new concepts introduced by Kamil or I didn’t faze him.

### Grant David <grd3> - Testing Team

Out of the rest of the team, I was most impressed by Grant - not by the amount of work he did, but the way he tackled challenges presented to him in a very can-do manor. He was always asking questions about how things worked and wasn’t phased when I asked him to program some Java for JUnit after proclaiming in the first meeting that he cannot program.

### Cellen Williams <cew10> - Testing Team

Cellen worked well within the team and helped create some really well written and laid out documentation during the time he was with the group, and I found that he was happy to ask questions about topics which I might have glossed over. However, I think that he would have been more productive if he took his initiative, as he often had the right answer to start with!

# Time Sheets

## Sam Sherar <sbs1>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| Researching VCS | 22/10/2013 | 1 |
| Setting up file structure | 29/10/2013 | 2 |
| Writing Project Plan Document | 22/10/2013 – 05/11/2012 | 5 |
| Spikework MySQL/JPA/Glassfish | 06/11/2012 – 14/11/2012 | 8 |
| Reviewing Testing document | 14/11/2012 | 1 |
| Component Diagrams | 04/12/2012 | 2 |
| Prototype for display | 19/11/2012 – 09/12/2012 | 8 |
| Integration and Testing Week | 28/01/2013 | 55 |
| Reworking Class Diagrams | 14/02/2013 | 2 |

## Kamil Mrowiec <kam20>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| Investigating Google App Engine | 19/10/12 | 3 |
| Discussing Server-Server API | 31/10/12, 07/11/12, 07/12/12 | 8 |
| Investigating DBMS | 02/11/12 | 3 |
| Improving UI Mockup | 05/11/12 - 06/11/12 | 10 |
| Working on Class Diagram | 25/11/12 | 4 |
| Creating sequence diagrams | 02/12/12 | 6 |
| Updating Class Diagrams | 02/12/12 | 2 |
| JPA Code for login prototype | 25/12/12 | 4 |
| Adding prototype friends list | 10/01/13 | 5 |
| Integration and Testing Week | 28/01/13-02/02/2013 | 45 |
| Reworking Class Diagrams | 08/02/2013 | 2 |

## Fiona Joseph Samy <fij1>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| Design Document | 22/10/12 – 02/12/12 | 15 |
| Group Meetings | October – January | 22 |
| Integration and Testing Week | 28/01/13-02/02/2013 | 45 |

## Andrew Watkins <ajw14>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| First Draft of the UI | 22/10/12 – 05/10/12 | 5 |
| Mapping | 29/10/12 | 1 |
| Group Meetings | October – January | 22 |
| Integration and Testing Week | 28/01/13-02/02/2013 | 35 |

## Grant David <grd3>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| Researching Testing Strategies | 22/10/2012 – 28/10/2012 | 2 |
| Rough tests for test document | 29/10/2012 – 04/11/2013 | 4 |
| Draft test document | 05/11/2012 – 11/11/2012 | 6 |
| Finalizing test document | 12/11/2012 – 18/11/2012 | 8 |
| Noting influences to Testing | 19/11/2012 – 02/12/2012 | 6 |
| Integration and Testing Week | 28/01/2013 – 02/02/2013 | 35 |

## Cellen Williams <cew10>

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours |
| Research sequence diagrams | 12/11/2012 – 18/11/2012 | 2 |
| Researching component diagrams | 19/11/2012 – 25/11/2012 | 3 |
| Reading over design spec and review | 26/02/12 – 02-12-2012 | 3 |
| Reading QA documents | 03/12/2012 – 10-01-2013 | 7 |
| Integration and Testing Week | 28/01/2013 – 02-02-2013 | 45 |

## Imran Mungul

|  |  |  |
| --- | --- | --- |
| Task | Date | Hours Spent |
| Group Meetings | 11/10/2012 | 2 |
| Group Meetings | 18/10/2012 | 2 |
| Group Meetings / Planning | 25/10/2012 | 5 |
| Group Meetings / Planning | 1/11/2012 | 4 |
| Group Meetings / Documentation | 8/11/2012 | 4 |
| Group Meetings / Documentation | 15/11/2012 | 4 |
| Group Meeting / Design | 22/11/2012 | 5 |
| Group Meeting / Design | 29/11/2012 | 5 |
| Design | 6/12/2012 | 4 |
| Design | 13/12/2012 | 4 |
| Group Meeting / Design | 20/12/2012 | 4 |
| Group Meeting / Design / Documentation | 27/12/2012 | 6 |
| Group Meeting / Documentation | 3/01/2013 | 4 |
| Documentation | 10/01/2013 | 3 |
| Design / Documentation | 17/01/2013 | 4 |
| Design / Documentation | 24/01/2013 | 5 |
| Coding Week | 31/01/2013 | 35 |

# Personal Reviews

## Sam Sherar <sbs1>

The group project was a big learning curve for me, even though I have been in situations before where I’ve lead a team, and was a lot of hard work at points, but in the end, the whole team worked well as a whole.

However, I did feel that group project as a whole didn’t live up to my whole expectation of what was described, as there was no incentive for the members: such as money or the open source “want” to develop, which made my job a lot harder to motivate and make sure that the work was being done on time.

The things that I will take away from this project are more life lessons than programming skills. These are such things like “If it’s not logged, it didn’t happen”, which is where we had problems with duplicated code in the project, as people were not assigning and closing issues quick enough.

## Kamil Mrowiec <kam20>

I am quite satisfied with the result of our work. Most of group members were doing their best, and I think the outcome is satisfying. No personality clasher had place, however, the main problem was the difference in capabilities and devotion between Sam and I, and other team mates. As Steve Jobs once said, “A players like to work with A players, they just didn’t like working with C players.”. Such situation occurred, unfortunately, what required us two to increase efforts, so that the product could also be called “A class”. Obviously, the aim of the project was to learn new things and improve existing skills, what I believe everyone achieved. All in all, I think the project was a very useful exercise, allowing us to meet team-work related problems in practice. Also, the technologies we learned will be useful in our careers, a fact I confirmed while sending IY applications.

## Fiona Joseph Samy <fij1>

I think the project went very well, even though things were moving quite slowly at first. Even though the Project and Deputy Project Leaders did most of the coding, everybody was given a task to do. At the start of the project, about half of the group (there were eight people in the group at that time) were in charge of the design specification and the other half were in charge of the testing specification.

I was part of the design team in the first semester. Therefore, I had to help write the design specification with three other members in my team. In the second semester, during implementation and testing week, I was made to do the JUnit tests and acceptance tests. I think the communication between the members of our group went pretty well overall. Even though not everyone showed up on time, we managed to get all our tasks done by the end of the day.

The biggest challenge for me was installing Glassfish. At first, I tried installing it on the university computer. But no matter how much I tried, I could not get it to work on it. So, I had to use my laptop instead. Even installing Glassfish on my laptop took a lot more time than it should have. But, I got it to work in the end, with help from my Project Leader, and I was therefore able to run the tests.

In the first semester, I felt a lot of people didn’t take the project as seriously as they should have. We even had one of the members leave, leaving us with only seven people in the group. The Project Leader did a very good job of managing the group. Whenever any of the group members were stuck with anything, the Project Leader always seemed to be willing to help, in spite of all the work he had to do. I think the main reason the project was so successful was because of the Project Manager’s ability to manage the group. He did a good job of motivating everyone to get things done on time, which ultimately led to the project being completed successfully.

## Imran Mungel <imm5>

The responsibilities I was assigned to do were to manage the design aspect for the Monster Mash game, this will give the team a clear objective to aim for in developing the game. I had to ensure that the members I was managing developed the necessary charts, diagrams and the required documentation that will go with it. My role was extremely crucial to the whole group, without any guidance there would be a lot of confusion resulting into a lot of time loss and chemistry between the team could be compromised.

The team was kept organised and I had ensured that we were on a steady progress pace.

I worked on the design factor of this project with my partners Kamil and Fiona, we had organised time outside of the scheduled meetings to tackle down any problems we came across, this developed our understanding of what we had to do much more easier. Due to us spending quite a bit of time together we developed a good relationship, this made communication between each other much simpler. We didn't really come across many problems in this project as we worked together, from my recollection we only came across very minor problems such as finding good or creative ideas for the application.  
  
The things I did well in the project was pretty much the majority of tasks involving the design of this game. I felt that I performed quite well and that the design part of this project was the ideal task for me, I feel that if I were assigned to do another task that I would not do as well since I don’t feel as confident in my programming skills etc.

If I were to do the project again, I would start my pre-studying for this assignment much earlier so that I could increase my knowledge in this field, saving me and my partner a lot of time. Another improvement I would say would be to be more punctual and attend more of the group meetings, this proved to be quite a struggle for me this time round.

What I have learned in this project is the understanding of github, and how it operates. I've learnt how to perform at my peak within a group as the key is down to communication. Sticking to the set deadlines proved to be hard, but after I succeeded the first time I had learnt to organise my time efficiently so I would never come across a situation where I would risk not submitting work again.

The skills I have learnt will most definitely benefit me in the future, as it will give me a sense of organisation, which is extremely crucial in the working world.

## Andrew Watkins <ajw14>

I did quite a lot of the very design parts, including the first draft of each page, and some of the final JSPs. I also wrote the JavaDoc for most of the classes. I felt I could have done some more coding than I actually did, especially back-end parts.

I felt that the initial design of the of the website was quite good in understanding a great deal of the project, even though it was very unlike the final design. Similarly creating the JSPs went just as well, and I don’t feel anything in particular I did went wrong particularly badly.

During this project I learnt quite a lot about how working in a group usually pans out, but the most important thing I learnt was that you can’t always do the thing you want to do.

## Grant David <grd3>

During the course of this project there are many skills that I believe I have either picked up or enhanced, the emphasis on teamwork and communication were apparent from the start as completing this kind of project without good communication between members would be next to impossible. I feel I have learned to more effectively communicate in a group environment to ensure that all pieces of the project fitted together correctly.

A difficulty I encountered while designing the test table was trying to make sure that there was a test for every part of the functional requirements present while also making sure that there are no unnecessary tests that were covered in an earlier requirement, there were many tests so this became confusing towards the end.

Finding issues for the project during implementation and testing week went well as we had a good system for logging the issues classifying them and assigning them to a group member to address via the github issues tracker so all problems were made apparent straight away.

Another issue I encountered was in fully understanding the inner workings of the program, as I have a limited experience with Java some of the code I found difficult to understand. This proved limiting when me and fiona were trying to create Junit tests for some of the methods. Luckily there were people on hand within the group to give an explanation of these

## Cellen Williams <cew10>

In all the tasks I was assigned in the group project, I feel I put in my best effort to complete them to the best of my ability. I feel that overall in the group project, I was not assigned many tasks, or assigned tasks which did not have as much responsibility as a few other members in the group, which I feel may have hindered me in many areas of the project.

Overall, I feel the group worked well together. I do, however, feel there were a couple of members of the group who did not pull their weight as well as the rest. I feel their contribution to the project was minimal, however they made themselves look like they were doing more work than they actually were by using another persons efforts to make it seem like they were using their full potential in the project, especially in Integration and Testing week.

Although I feel I performed each task to the best of my ability, I do, however, feel I could have asked the project leader more often, to be assigned tasks with more responsibility to improve my performance in this project. I felt that I was only assigned one or two tasks with a higher responsibility.

Overall, I think our group has worked well together, however, I do feel there were a couple of people who could have put in much more effort into the project, especially in Integration and Testing week. I feel that a couple of people were using other peoples efforts to make it seem like they had done the work they were assigned, and another group member palmed off their assigned work to other people to complete, rather than just trying to complete their assigned work.

# Change log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF Number | Date | What Changed | User |
| 1.0 | N/a | 31.01.2013 | First Draft | cew10 |
| 1.1 | N/a | 10.02.2013 | Filled in and proofread | Sbs1 |