**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 17

3. REFERENCES 22

4. DOCUMENT CHANGE HISTORY 22

# 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.

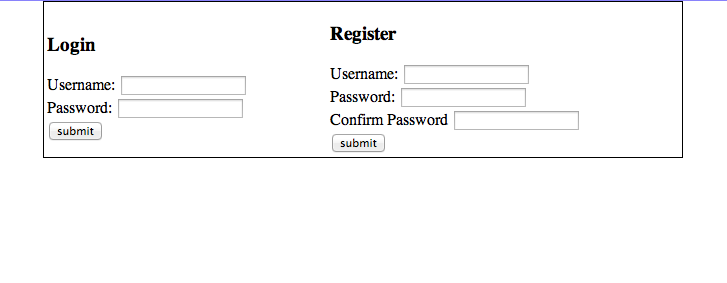


Fig 2 - Registering and Logging in

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.

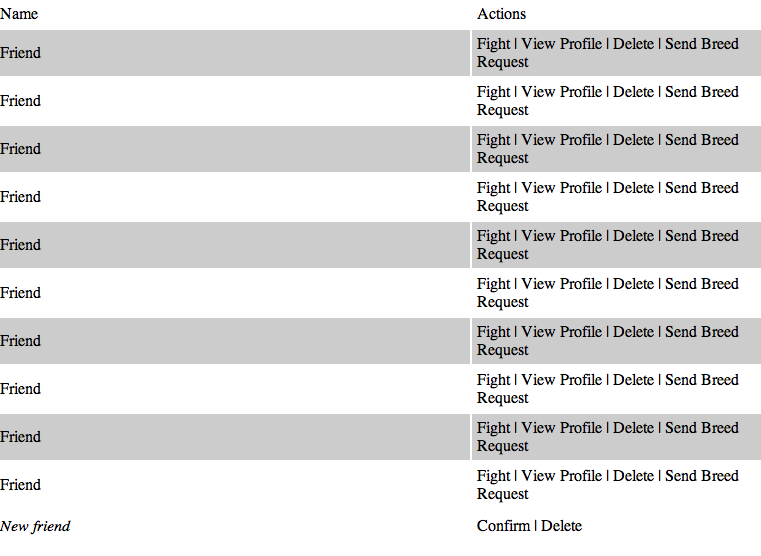


Fig 3 - Friends list

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.

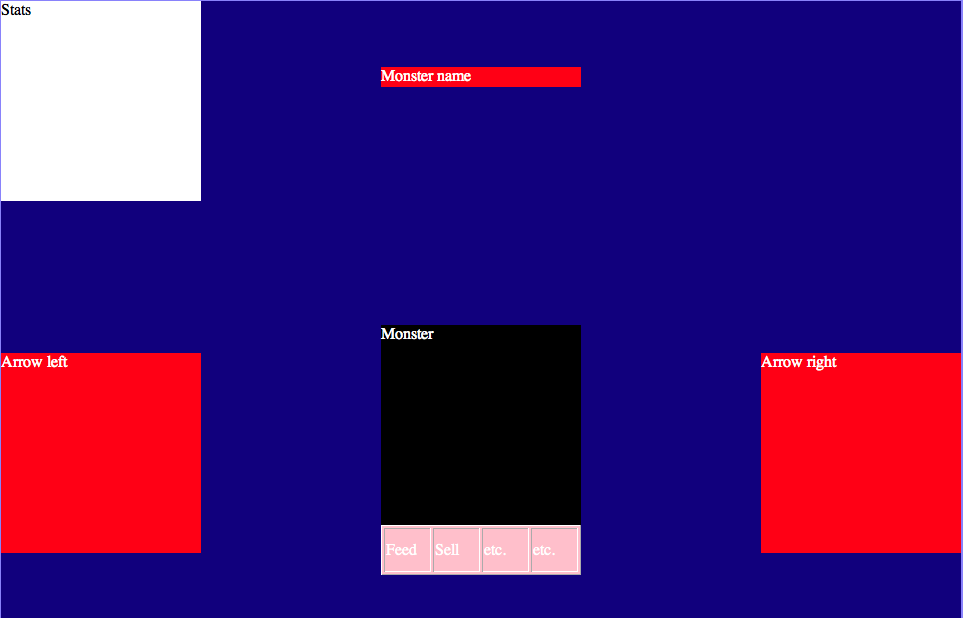


Fig 4 - Monster Information

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.

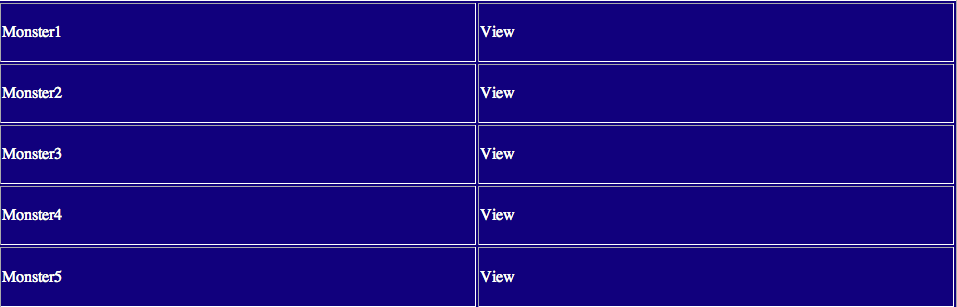


Fig 5 - Stable list

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

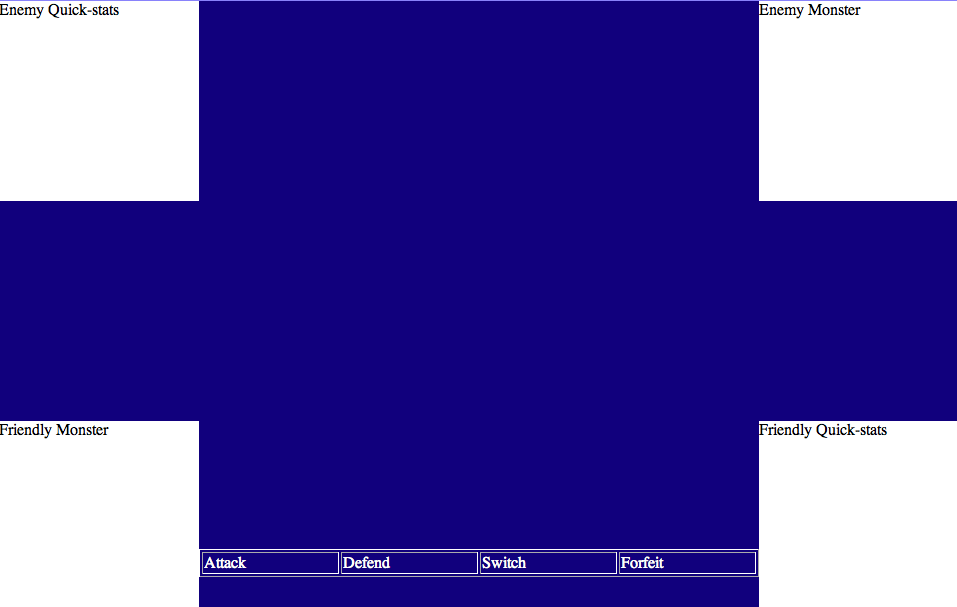
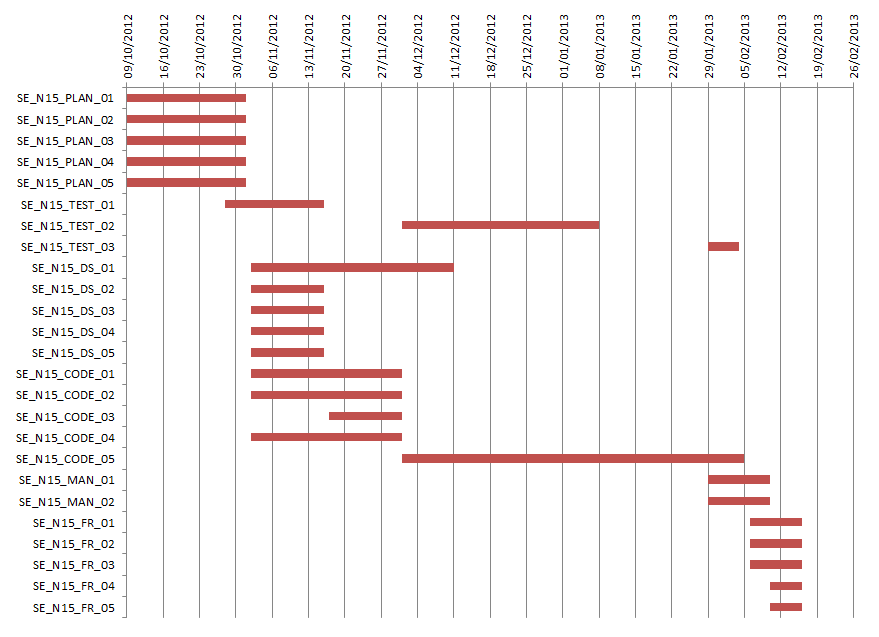


Fig 6 - Fight screen

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

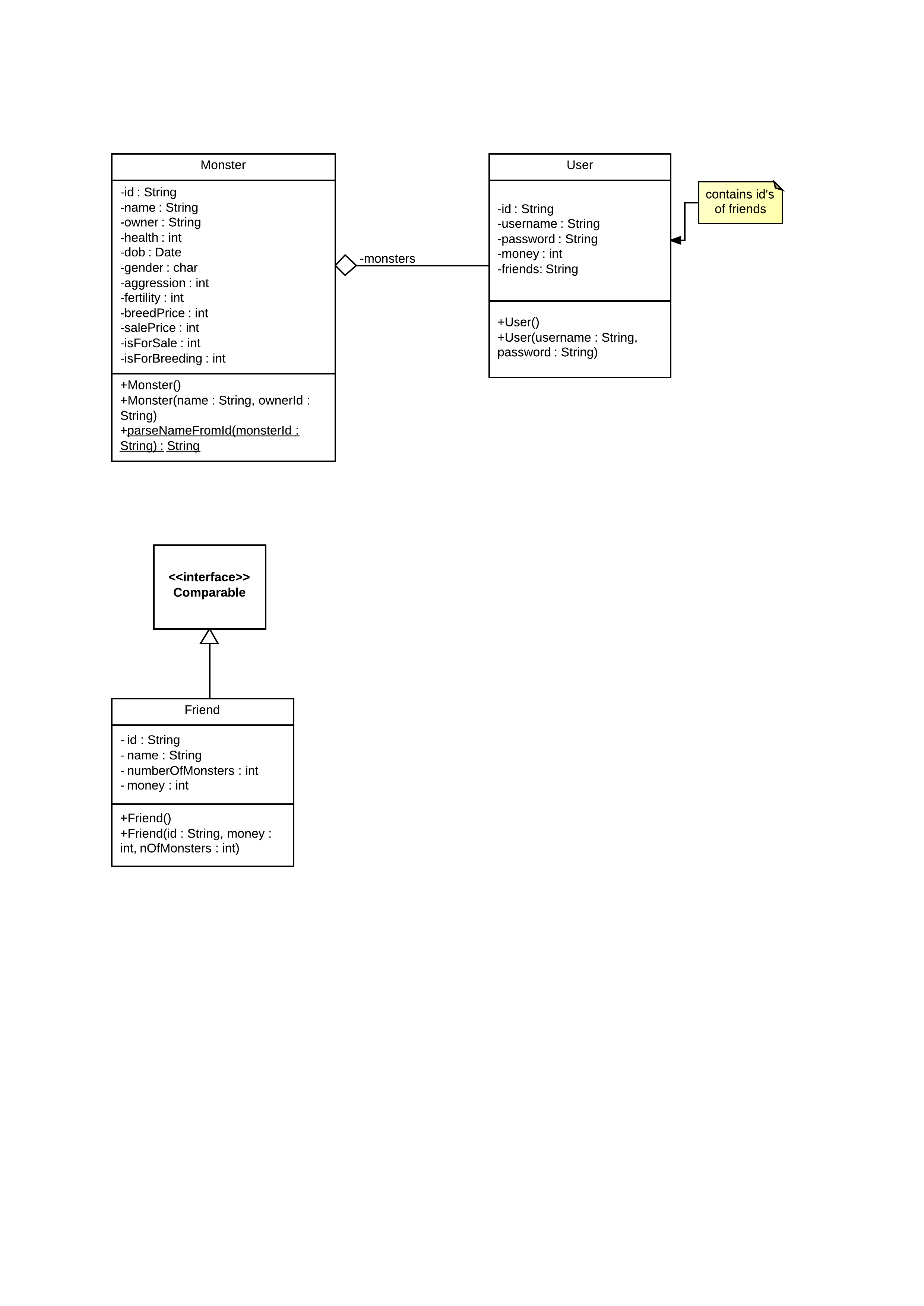
This document shows all the diagrams, methods and descriptions regarding the design factor of this project. It reveals how the diagrams all inter-link to each other from the server side to the design. The descriptions of the classes used are clearly explained in this document, as well as the methods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

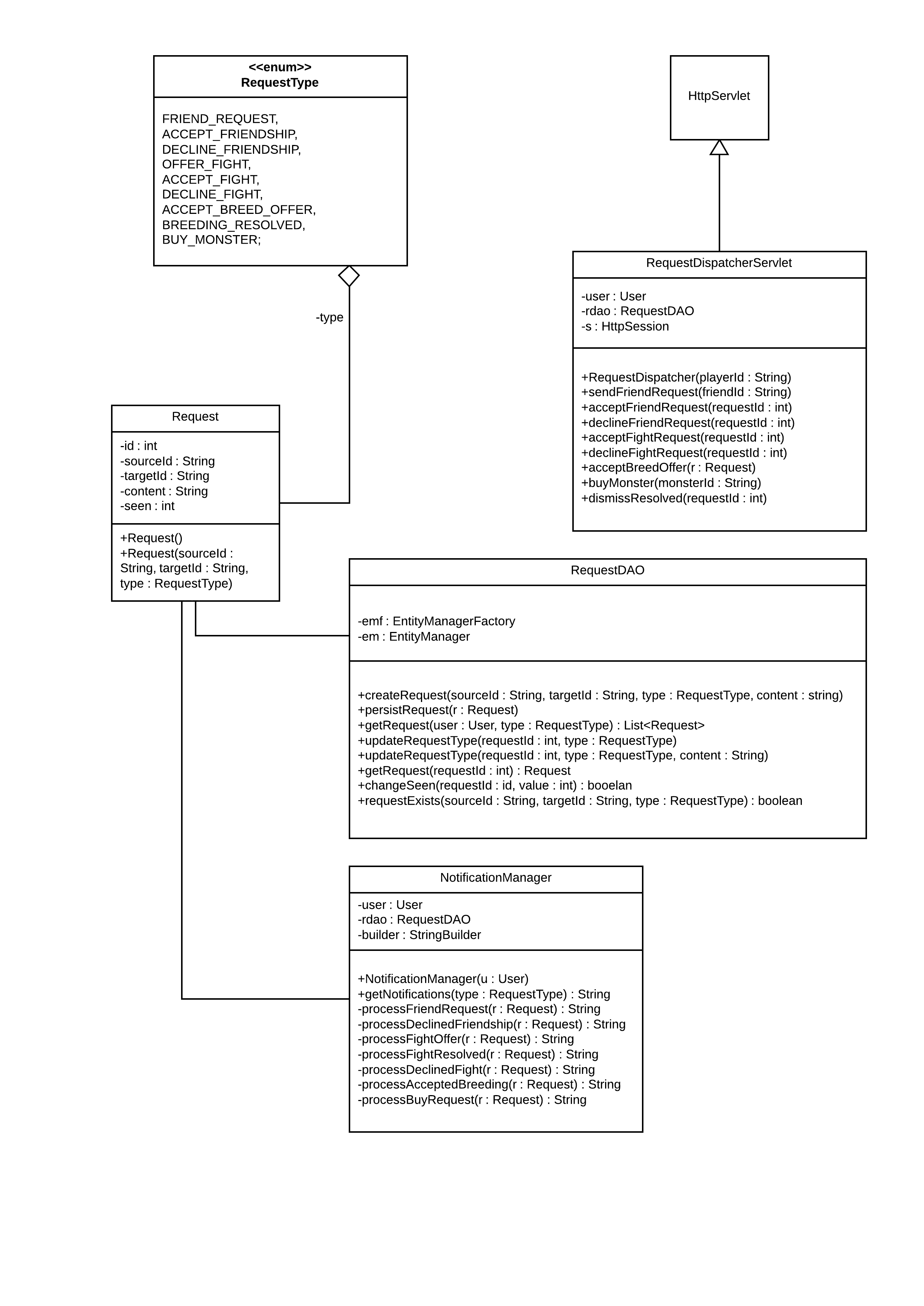
### Objectives

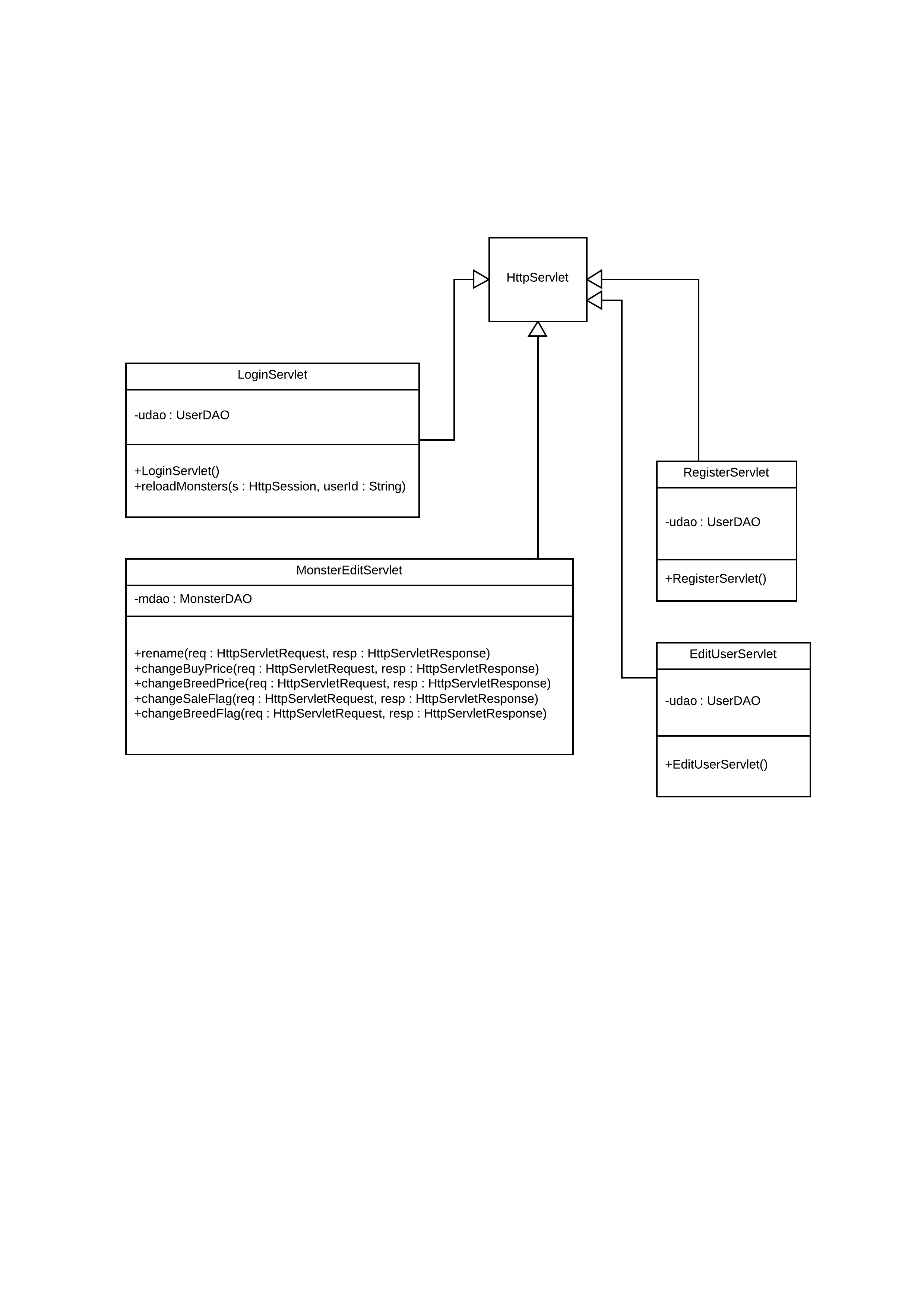
The priority objective for the design team was to create the necessary resources for the application to the point that it worked efficiently, as well as having an impressive appearance. The software structure, components, interfaces and the data revolving around the design factors would all have to be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

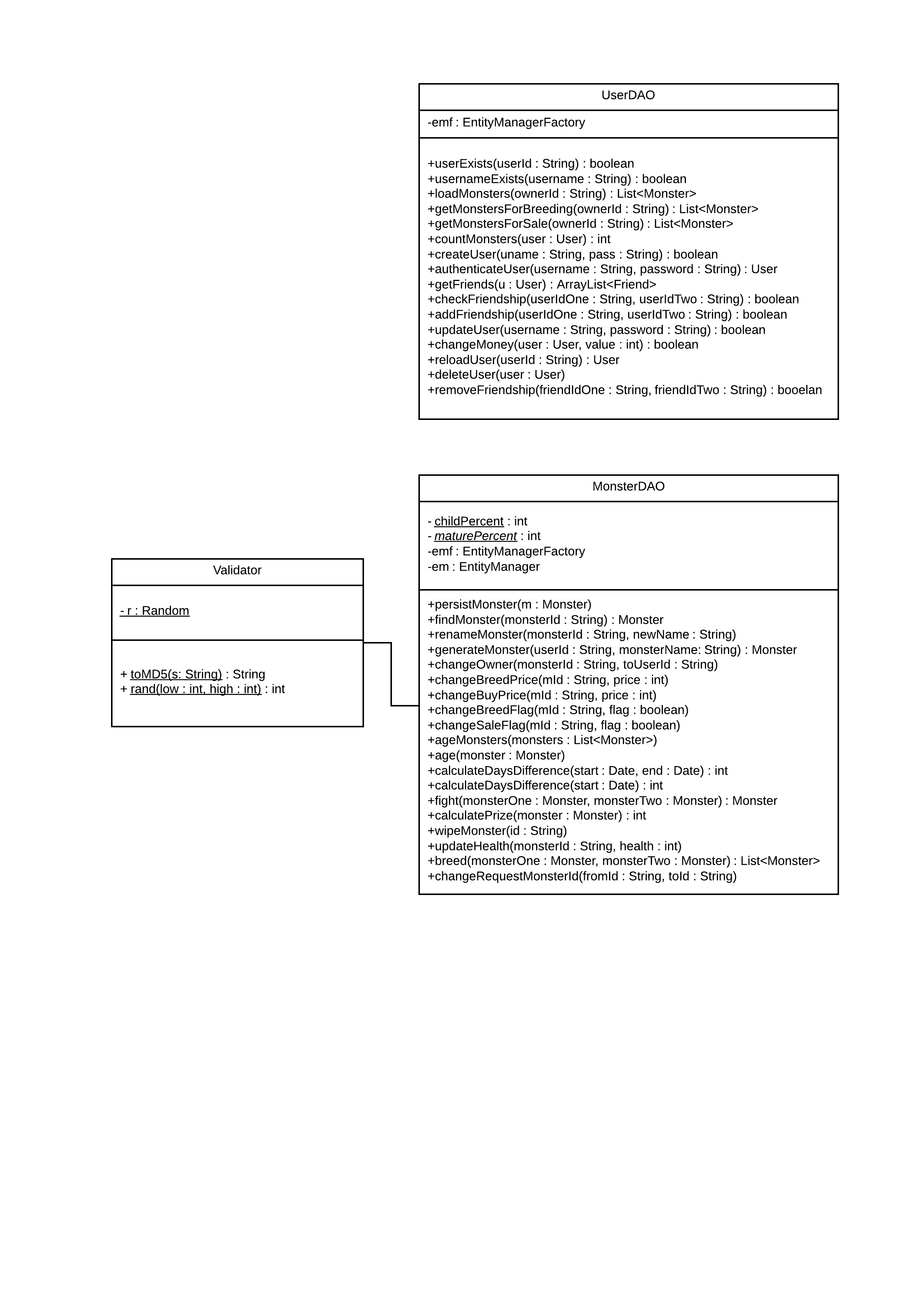
## Decomposition Description

### 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 |

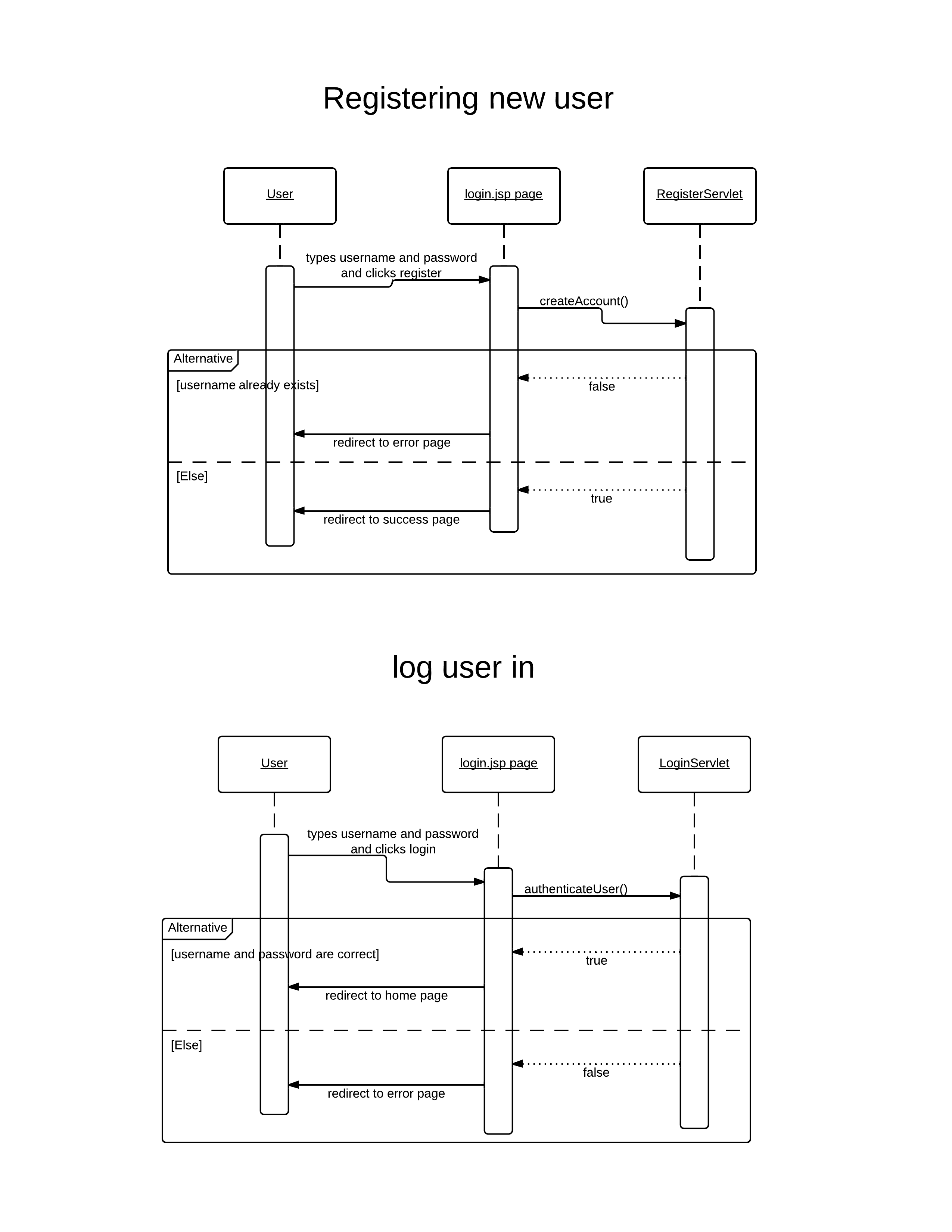
## 

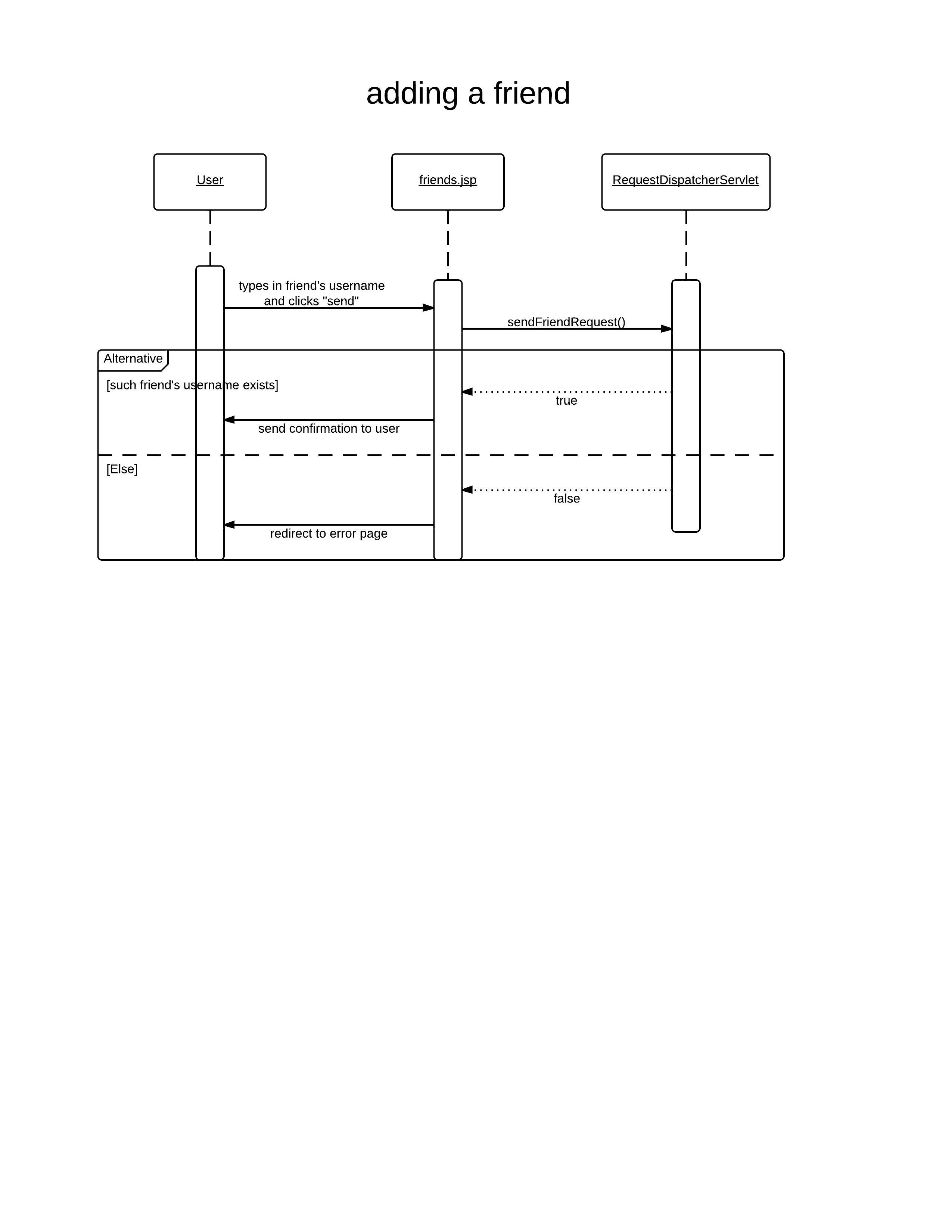
## Dependency Description

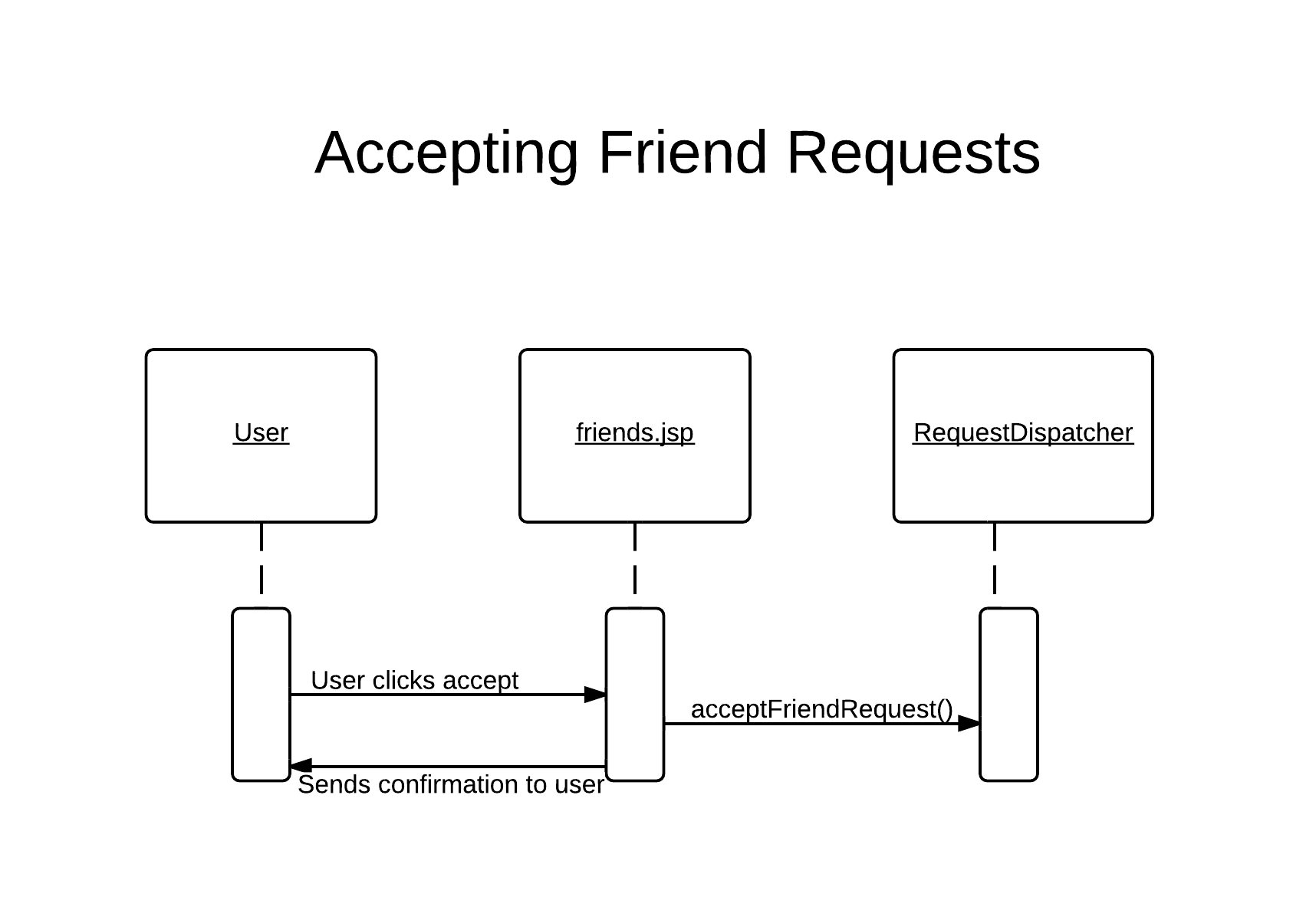
### Component Diagrams

## 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 |