Social Battle Documentation

Project for Service Technologies 1 Course

Lorenzo Affetti 799284

Contents

Chapter 1

Introduction

Before talking about the project in its details, I think it is worth to spend a few words about the general requirements imposed by the nature of the Service Technologies Course from which the project comes from, also keeping in mind that the final application will be used to test the performance of a IaaS system.

Given this, I will have to fulfill three specific requirements:

- The system will have to expose some documented services through an API;
- The system has to integrate other services;
- The system has to be computationally intensive, both for the database and the CPU.

1.1 Project Description

The project aims at mixing the features of a social network and a role-playing game into a web application named *Social Battle*.

To better define the context, we specify the main features of the two elements:

- **Social network**
  The main features of a social network are the relationship with other users and the interaction with them.
  The relation could be a friendship (a symmetric one) or a fellowship (an asymmetric one).
  The interaction among (or between) users could be chosen among posts,

instant messaging, deferred messaging, votes, comments, tags and others.

- **Role-playing game**
  The central idea of a role-playing game is the growth of the character (or the team) we are using, in terms of the level of experience and the equipments and objects in general.
  There are a lot of RPGs (especially the one played exclusively on-line) in which there is no other goal than skilling more and more your main character.

Social Battle will take most of the features of a social network adding the ones of a role-playing game.

Thinking of *Facebook* as our landmark for the social networking part, you can find below the matching between some *Facebook* components and what they might become in *Social Battle* context:

- **The page**
  Something as a location. A place in which a user can fight mobs.

- **The friendship**
  It could become an asymmetric relationship, suck as *Twitter*'s fellowship. Once *User A* follows *User B*, he could follow all the progress of *User B*;

- **The post**
  Could replace loots.
  The *like* on a post could become the grabbing of a loot.

- **Instant messaging**
  The fight between a user and a mob (PVE) or between two users (PVP).

These were only examples to better understand what *Social Battle* could become.
As you may have already understood, the system will not be provided with a GUI except from the one that usually a social network gives, and so a textual one.

1.2 Glossary

I think it is worth to define some common words that will be used in this documentation.

- **SB**: *Social Battle*, the system we are talking about.
- **RPG**: Role-playing game.
- **Mob**: An enemy controlled by an artificial intelligence (maybe the server).
  The user will have to fight against the mob to obtain experience (money, objects, equipment) and to skill his character.
- **Spawn**: The act of "appearing" of a mob.
  The idea is that, if the user is at a location (a web page in our case), mobs will appear randomly and the user will have to fight them.
- **Guil**: Term borrowed from *Final Fantasy*.
  A guil is the virtual currency of the RPG. Generically "money".
- **Loot**: The items dropped by a mob when it has been defeated by the user.
- **EXP**: The experience points earned by the user after defeating a mob.
- **Equip**: Abbreviation of "equipment".
- **PVP**: Player Versus Player fight.
  A fight between to users.
- **PVE**: Player Versus Environment fight.
  A fight between a user and a mob.
- **Ability**: The skills of the character.
  Social Battle will give the possibility to use the classical RPG kinds of abilities, and so physical and magical.

1.3 Goals

To identify the goals, we have to think of what our system has to provide both to the end user (in terms of a web application) and both to the developer (in terms of the API).
My aim is to develop both the API and a web application that consumes them.
For now, when I talk about goals, I mean the goals of the application that the end user will use.

So, the final system should give the user the possibility to, at least:

- Register and log in to the the system (possibly using a pre-existent social network).
- Skill his character and so fight against mobs (maybe other users) to gain EXP, equipments, loots and guils.
- Follow some characters to learn about their progress and be followed by other users in turn.
- Edit his personal information.
- Explore the world around him to meet new players and to fight against different mobs.
- The system should always be a social platform, so the user should be able to interact with other users.
- Share his results (about battles, earned badges or others) on a social network.

It is obvious that the API should be thought with the purpose of making it possible to the developer (which in this case it is me) to build a web application that fulfills the goals explained above.
In later chapters a detailed explanation of the API will be provided.

1.4 Actors

The Actors in our system are only two:

- **The guest**
  The unregistered user which can only sign up to *Social Battle*.
- **The user**
  The registered user, which is only a guest which has signed up.

The topic about which action an actor can perform in the system will be deepened in the *Use Case* section.

Chapter 2

Requirements and Specifications

In this chapter the requirements for our web application (the one the end user will use) and for the APIs (that a possible developer will use) will be listed.

It is important to say that the requirements that will be listed below, are the theoretical requirements of a *full* application.
I will write a paragraph in which I will specify the minimum functionalities of a working application.

So, from the goals written in the previous chapter we can derive our general requirements.
I will write down here, for each goal the requirements that will fulfill it.

- **Register and log in to the the system (possibly using a pre-existent social network).**
  - Provide a sign up functionality (at least with *Facebook*).
  - Provide a sign up functionality not using an already existent social network.
  - Provide a sign in functionality.
- **Explore the world around him to meet new players and to fight against different mobs.**
  - Provide something like a "location" in which a player can simply chat with other players and find mobs.
- **Skill his character and so fight against mobs (maybe other users) to gain EXP, equipments, loots and guils.**

  - Provide the possibility to fight against a spawned mob according to the abilities and skills of the character.
    This means that the server shouldn't spawn mobs that have a level too high compared to the one of the player.
  - Every time a player defeats an enemy the system should reward the player giving him EXP, guils and so on.
  - There should exist a place in which the user can manage the status of his character in terms of the equipment, the abilities in use and the items collected.
  - Provide a PVP functionality.

- **Follow some characters to learn about their progress and be followed by other users in turn.**
  - Provide the possibility to a user to follow another one.
  - If *User A* follows *User B*, the system should properly notify *User A* about *User B*'s progresses.

- **Edit his personal information.**
  - Provide a "edit profile" functionality in terms of the avatar and other personal information. Not about character's skills.

- **The system should always be a social platform, so the user should be able to interact with other users.**
  - Let the user post and comment. This functionalities will have to be well integrated with the aim of the system. The post and the comment will be an extra to skill the character.

- **Share his results (about battles, earned badges or others) on a social network.**
  - Provide the possibility to share (using the social network which the user signed in with) his progresses (defeating a mob, earning guils, earn equips or earn badges).

2.1 Functional Requirements

From the general requirements wrote above we can define the functional requirements and so the functionalities that the system has to provide to

users. A guest can only sign up to the system; a user, instead, can exploit the entire set of features provides, so:

- Sign in;
- Modify his profile;
- Explore locations;
- Change equipments;
- Add/remove abilities;
- Manage items;
- Follow/stop following another player;
- Share his results;
- Fight against a mob;
- Fight against a player;
- Post;
- Comment;
- Sign out.

2.2 Non Functional Requirements: the API

First of all the APIs to *GET* and *POST* data from/to the system will be necessary. So, in general, services as the sign up, posting, comment, *like* and other social facilities will be provided.
The same thing stands for the peculiarities of the platform, such as, activate an ability, use an object, collect an item and so on. In general, all the possibilities to modify the persistent data will be exposed.
To built this part of the back-end of the application, I will get ideas from *Facebook's Graph API* which I think it is a great model to build a well organized exposed API.

In addition to this general services, I think it is worth to provide a *Battle Service*, to let any system include something like a *widget* that will include a full *SB* battle. In this way, any system will give the possibility to a *Social Battle* user, to face random mobs even outside *SB* itself.

It would be great if we also give the possibility to fight against a custom mob (in addition of the widget). Thanks to this anyone who includes the *Battle Service* will be able to advertise his brand/company through mobs and enrich the game itself. For example *Facebook* could think of the "Mark Zuckerberg" custom mob; *Coca Cola* the "Polar Bear" mob; *Stack Overflow* the "Bad-Code" mob and so on.

Further considerations about the APIs will be made in Chapter 5.

2.3 Specifications

Here are some further specification to make better understand how the system will work:

- The "location" should be something like a *Facebook* page, a "Room" with its own features. For instance in a specific room a user can find some mobs and some others not.
- Once a user is in a room, mobs will be spawned randomly.
- The battle will be an ATB (Active Time Battle) one. So, any attack (ability) to be performed will require a certain specific time to be performed. Once the user selects an ability, he will have to wait for the time required. After that time passes, the ability will be performed and the user will be able to choose another ability; the battle goes on following this steps.
- The EXP and guils gained from a battle have to be proportional to the power of the mob.
- When mobs are defeated, they will leave items (loots, objects and/or equipment) and guils to the winner.
- If a user looses a battle (i.e. he dies), it will not be "game over". There will be a penalty because of the KO. The penalty could be applied to guils or to the EXP.
- Some rooms in which a user cannot fight mobs will be provided. These rooms (we will call them "relax rooms") will be locations in which a user can chat with some artificial intelligence to buy items; post freely to interact with other users and, maybe, exchange items with them. There would be also PVP locations ("Arenas") in which a user can start fighting (through a chat mechanism) against another user.

2.4 Basic functionalities

I think that it will be very difficult to develop the system described above in few months. So I will shrink the features in some basic one. I will list down here the functionalities which will be realized and which will give a minimal working application:

- Sign up and sign in;
- The battle (against mobs), which is the core of the application;
- The spawning of mobs;
- The post and the comments;
- The following mechanism;
- The rooms;
- The possibility to skill the character (at least equipment and abilities).

Chapter 3

Scenarios

In this chapter you will find the most important scenarios among the entire set of possible scenarios that a user could face in interacting with the final application.
I decided to list only the most important, because it happens very often to list scenarios which are self explanatory and so they are only a waste of time to read and to write.

I think that the scenarios that needs some clarification are:

- The Log In
- The Fellowship
- The Rooms
- The Battle

The subject of our scenarios will be John who heard about *Social Battle* by a great friend of him, Jill.
This friend told him that the social network is highly addictive, so John decides to give it a try and sign up.

3.1 The Log In

John enters the home page of *Social Battle* and discovers that he can decide between three options: sign up with *Facebook*, with *Twitter*or without using any social network. John decides to sign up without any of the specified social networks, so he enters a nickname; his e-mail address and types his new password (he also retypes it for security reasons).
Now John has subscribed to *Social Battle* and he can use it.

First of all, John enters his profile and decides to edit it because he wants to add a new avatar. So he enters a new avatar and, watching at the "edit profile" page, he discovers that ha can bind his account to *Facebook* and to *Twitter* if he wants, even if he didn't do so in the sign up phase. John does it, so he performs the login with the two social networks.
Social Battle informs John that, doing so, he will be able to share his results with his *Facebook* and *Twitter* friends.

3.2 The Fellowship

The first thing that John wants to do, after editing his profile, is to search for Jill. So he types Jill's full name in the search bar and opens her profile. Jill is using *Social Battle* for weeks, so his character is already at level 16 (John's one is level 1) and has a lot of abilities and items equipped which John doesn't know about. John decides to click on "Follow", in this way he could follow all Jill's progresses, such as levelling up, new mobs killed, new items collected and so on.

3.3 The Rooms

Some weeks have passed, John is now a veteran. As any other day, John logs in to *Social Battle* and enters "The Shop" room. This room provides the possibility to buy items from an AI controlled merchant. John is out of "potions" and so he opens a chat with the merchant and types "\buy potion". Doing this, John looses 125 guils, but gains a potion in his inventory.
Now John can go to skill his character and so he enters his favorite room: the "Thunder Plains". In this room he can find his favorite mobs: "Iron Giant" and "Qactuar". In fact, after a few seconds, a wild "Qactuar" spawns on the bottom of his screen and the battle starts.

3.4 The Battle

The "Qactuar" always performs his strong attack "1000 Needles" which inflicts 1000 points of damage. John knows that he has to kill it rapidly so he starts repeatedly attacking it.
As just as John types "\" a drop down menu appears downside the input bar containing all the possible abilities that his character can perform. John proceeds in typing "att" and the drop down menu reduces to suggest "attack". John presses "Enter" and the physical attack is selected. The ATB

bar charges and, once it has arrived to full charge, the attack is performed and "Qactuar" looses 242 points of damage.
After a few attacks the mob dies and John gains a "Speed Sphere" (item) that is added to John's inventory, 350 EXP and 48 guils.

After killing the "Qactuar" John looks at the "Thunder Plains" page. In the center of the page he can find the loots discarded by other users at the end of a fight. Clicking on "grab" John takes other two potions.

After this fight John is told by Jill that *Whatsapp* is giving the possibility in its official site to *Social Battle*'s user to fight a custom mob. John goes to *Whatsapp* site and on the top right corner there is a widget. John clicks on it and, after logging in, the battle against "Billions" mob starts.

Chapter 4

UML Diagrams

4.1 Use Case Diagram

The Use Case Diagram is needed to better understand which actions any actor can perform in the system and the dependencies between them.
For a better comprehension of the diagrams we will separate the diagrams per actor and per functionality (in the case of the registered user).

4.1.1 The Guest

The guest use case diagram is simple, because he can only register to the system:

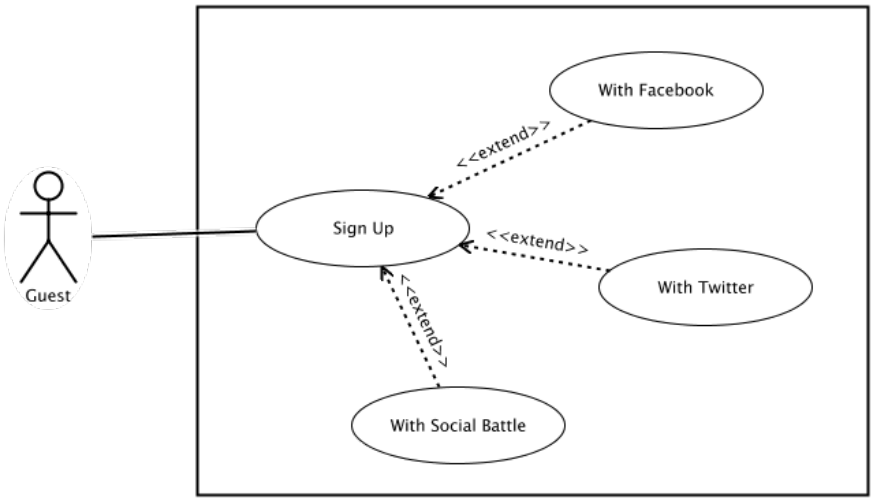

Figure 4.1: Guest Use Case Diagram

4.1.2 The User

In the case of the user, things become more complex. Separating the different set of actions will help in understanding the use case diagram:

- **Sign in**

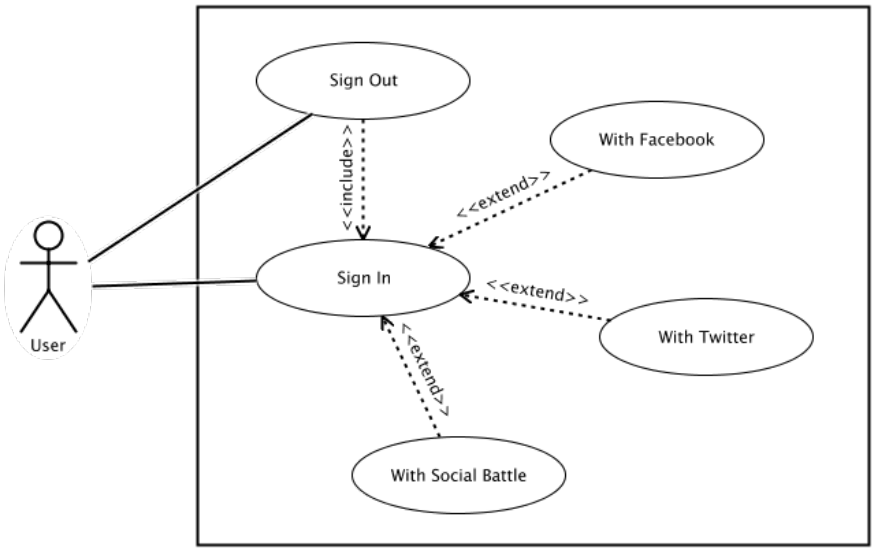

Figure 4.2: User sign in and sign out Use Case Diagram

- **Social . . .**
  Any use case pointed by the actor *includes* the use case "Sign in".

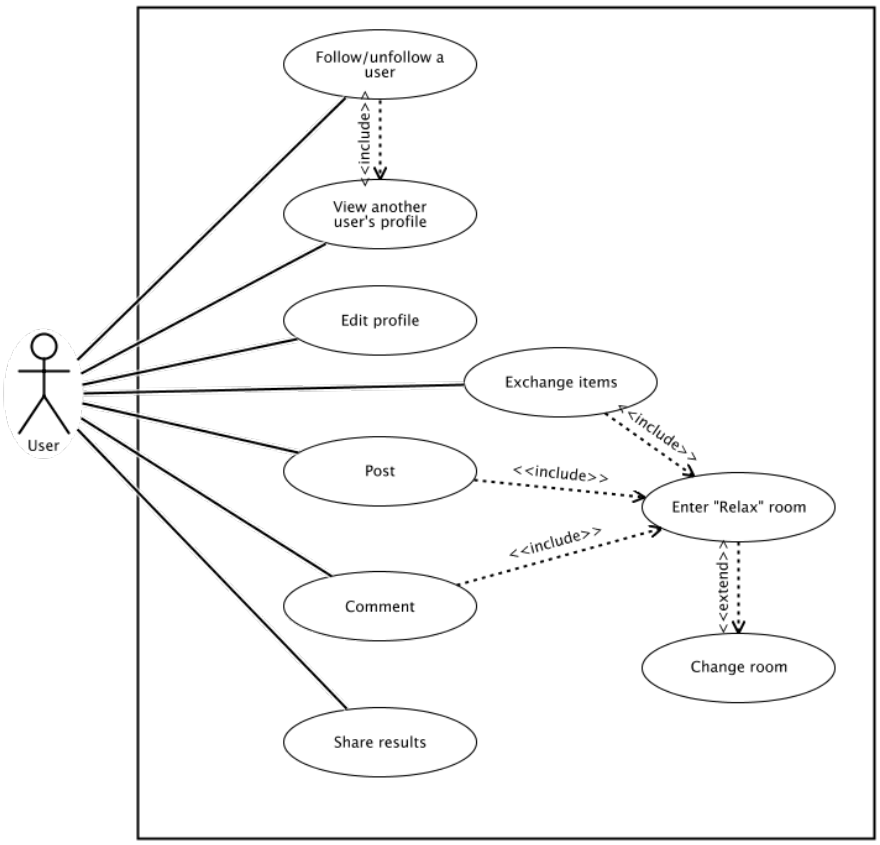

Figure 4.3: Use Case Diagram for the social features

- **. . . Battle**
  Any use case pointed by the actor *includes* the use case "Sign in".

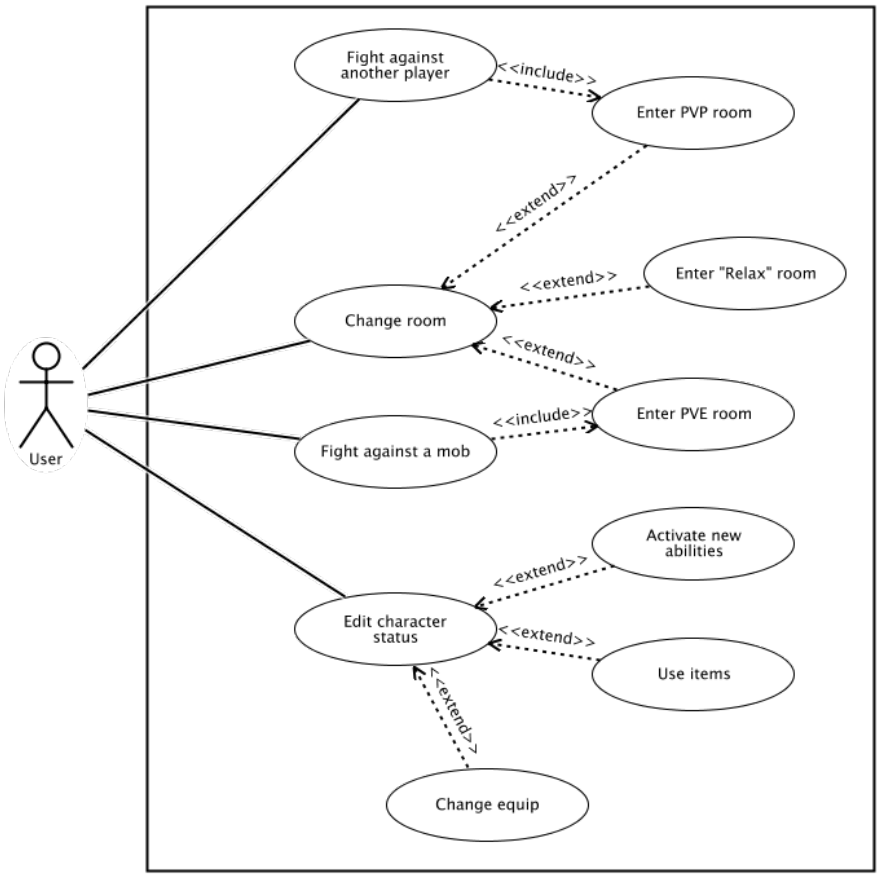

Figure 4.4: Use Case Diagram for the "battle" features

4.2 Class Diagram

Eventually, to better understand the entities involved in our system and the interactions between them, we draw a simple class diagram:

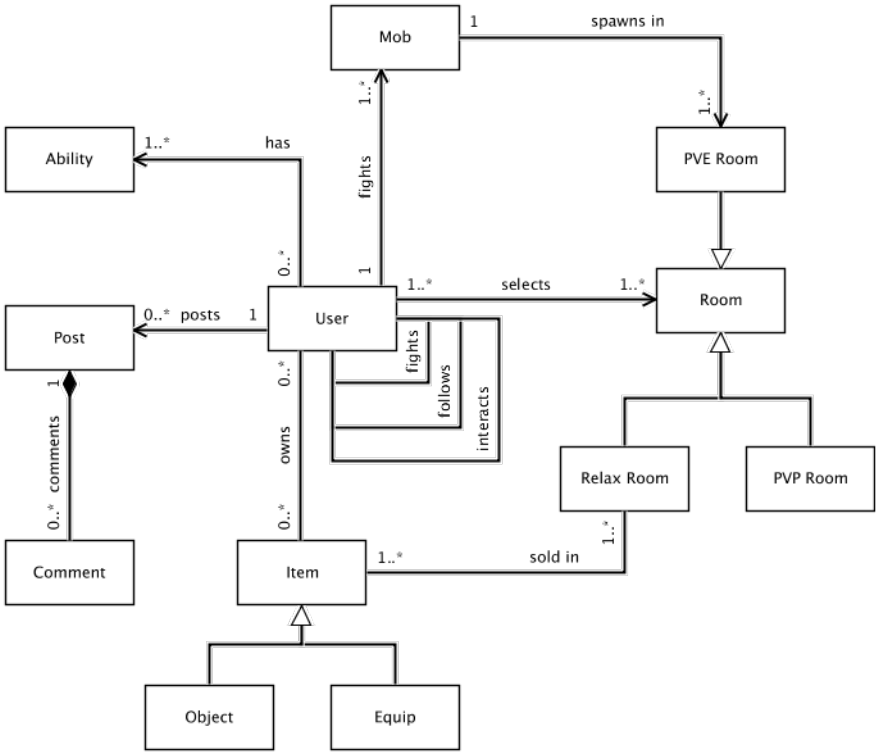

Figure 4.5: Class Diagram

Even though the class diagram is complete, it cannot clarify the correspondence between the action that a user can perform and the room in which he is able to perform them.
Even though this can be acknowledged analyzing carefully the use case diagrams, I think it would be useful to provide a state diagram relating rooms and the action that a user can perform in each of them:

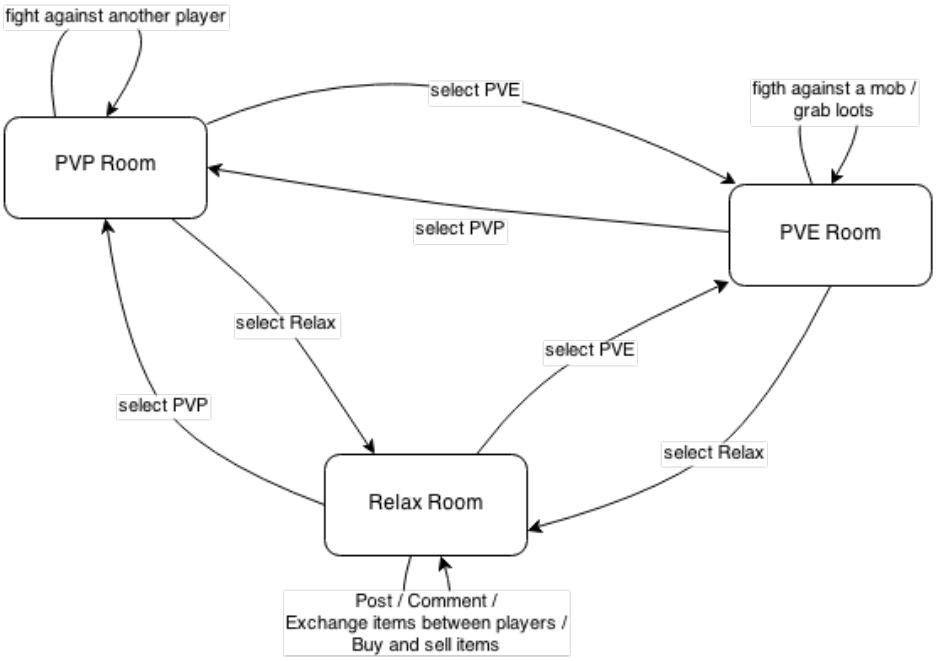

Figure 4.6: State Diagram for Rooms

Chapter 5

Architecture Description

Before making any consideration about the architecture of the system we have to separate the two applications that will be made, the end-user application and the back-end.

5.1 The Back-end

The back-end is intended to be used by developers that wants to use *Social Battle*'s services to develop an end user application. Its architecture will be a SOA (Service Oriented Architecture) for two main reasons: the first one is because of the nature of the course this project is done for; the second one is because SOAs have some features which have lots of pros. For instance, services are independent, loosely coupled, can be distributed with low effort. For this reason a SOA is also easy to maintain and it is easy to evolve.

Our back-end will offer basically three services:

The Battle Service : A service which allows to start a battle and to finish it; to retrieve active monsters in a room; to grab loots. Additionally, this service, will give the possibility to include an entire battle (a PVE or a PVP one).

The Social Service : This service will be the social one, and so it will provide the possibility to retrieve user information, to post, to comment, to follow and other similar functionalities. It will also provide the possibility to share results using *Facebook* or *Twitter*.

The Sign in/up Service : This service will provide he possibility to sign in, up and out the system (possibly using some security policy).

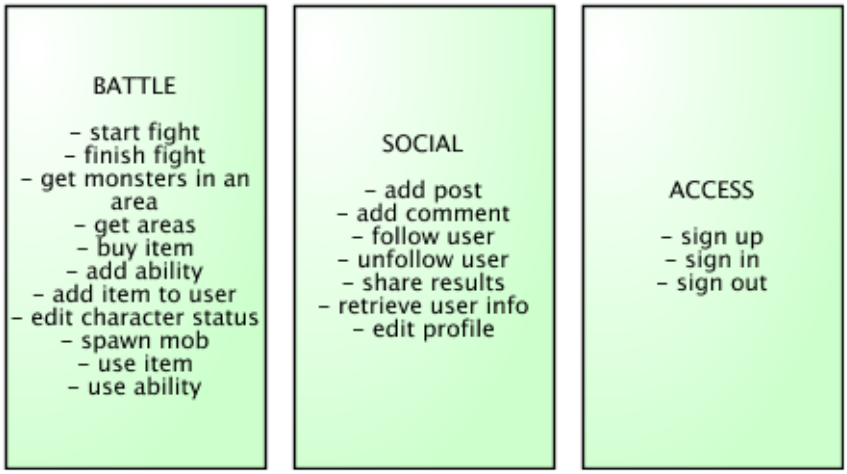

Figure 5.1: Scheme of the Back-end System

5.2 The Front-end

The front-end, on the other side, will be a simple client-server application. This application will exploit the API exposed by the back-end. We report here a diagram identifying subsystems of the application and their interaction with the back-end:

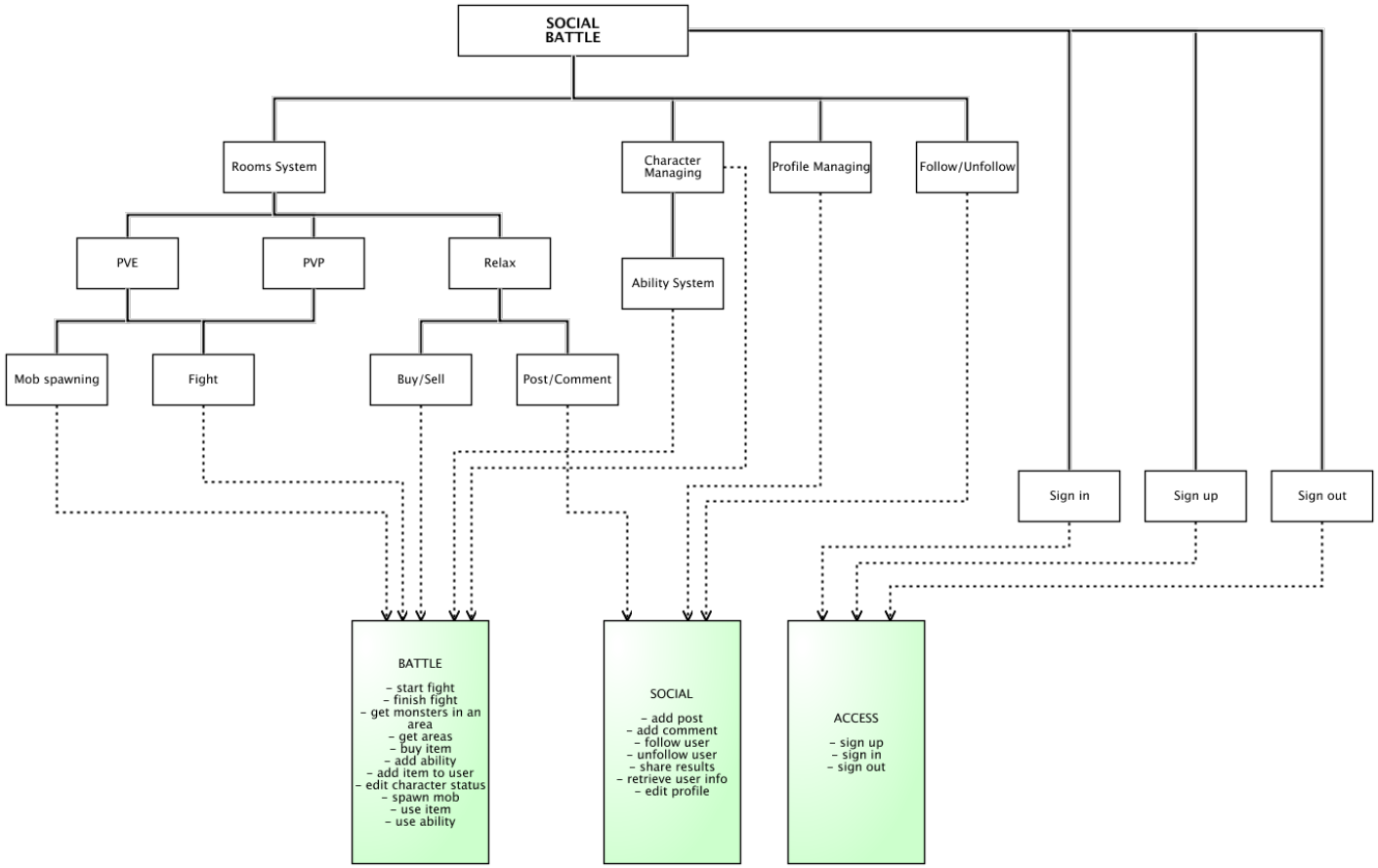

Figure 5.2: Scheme of The End-user Application and the interaction with the Back-end

5.3 Technological Considerations

The back-end will be developed using *Django framework* which already provides a working (WSGI) web server and a ORM to interact with the database. The database will be a MySQL one, and so, a relational one.
I decided to use, united with the SOA architecture, a REST architecture. To better achieve this I will use *Django REST Framework* on top of *Django*.
We will need a service to handle asynchronous real-time notifications (at least for the automatic spawning of mobs). For this I will use *Django Announce*, another framework which lays on *SocketIO* and a *NodeJS* server. This framework is still in "alpha" state, I hope not having any problem.
For the security of the authentication *OAuth2* protocol will be used.

If I will decide to implement a chat mechanism, I will have to decide if relying on *Django Announce* or building an ad-hoc a *SocketIO* server that will "talk" with a *Twister* server. This seems to be a common practice to handle chat issues.

The back-end will have to integrate *Facebook*'s and *Twitter*'s services. To achieve this purpose I will use some famous SDK written for python language: *Django Facebook* and *Twython*.

In addition to this I think I will use *Gravatar* for the auto-generation of random profile images and *Trianglify* for the auto-generation of backgrounds.

Chapter 6

Persistent Data Management

6.1 Conceptual Design

This section contains only an Entity-Relationship Diagram, which gives a quick overview of how entities will be organized as persistent data.

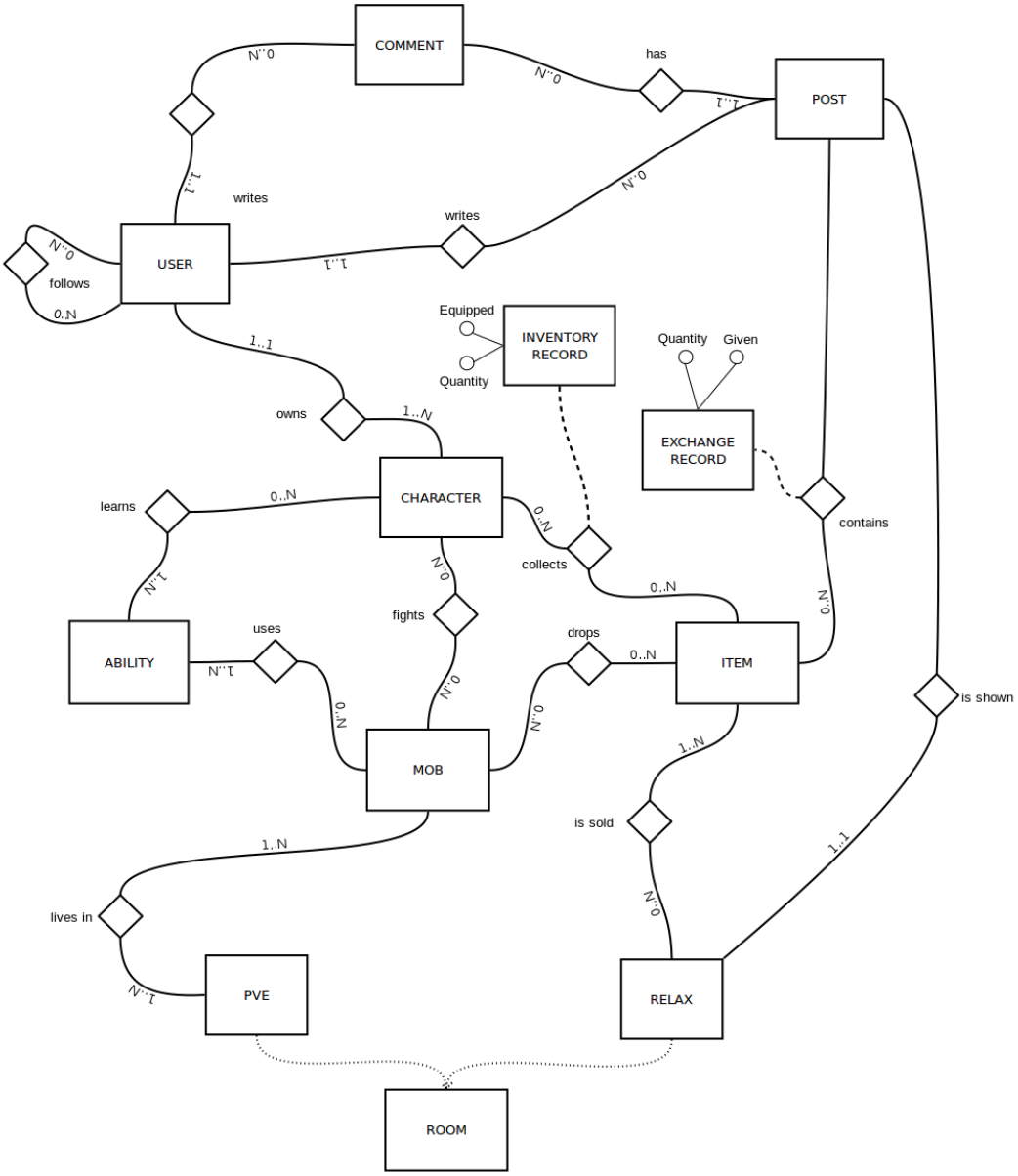

Figure 6.1: Logical ER Diagram