|  |
| --- |
| **IMPLEMENTED SECTIONS** |
| Pokedex JSON |
| PokeBattle Features  -Destroy a pokemon and give all its exp to a pokemon of the same type  -Revive a fainted pokemon with berries caught in poke catch mode  -View all of a player’s pokemon; view detailed information about a particular pokemon  -Level up and evolve pokemon  -Battle between two players; server can support multiple battles |
| PokeCatch Features  -Automove for 120s  -Catch wild pokemon using pokeballs; collect berries and pokeballs at pokestops  -Spawn pokemon at random locations every X mins and despawn after Y mins; replenish pokestops with either a pokeball/berry at A mins, these disappear if uncollected after B minutes |
| Others  -Integrated the authentication server and the battle server using MutiService  -Authentication module with signup, login, logout, open authentication using access token |

Term Project – PokemonGo inspired

Net centric programming

ThuyPham ITITIu13171

2017

Contents

[1. Architecture 1](#_Toc483646773)

[2. Data storage and access 2](#_Toc483646774)

[2.1. Data stored in JSON files and .TXT files 2](#_Toc483646775)

[JSON format 2](#_Toc483646776)

[TXT file 2](#_Toc483646777)

[2.2 Data in classes 3](#_Toc483646778)

[2. Sequence diagram 3](#_Toc483646779)

[2.1. Authentication Module 3](#_Toc483646780)

[2.2. Battle Module 5](#_Toc483646781)

[Features and their sequence diagrams 5](#_Toc483646782)

[2.2. Catch Pokemon Module 5](#_Toc483646783)

[Features and their sequence diagrams 5](#_Toc483646784)

# Architecture

The ***client-server model*** was used in the writing of all modules.

The use of the twisted python library in writing the authentication and battle modules( server and client) means that the ***asynchronous event driven framework*** was also used in the project.

I chose to use the event-driven architecture and asynchronous model because it optimises the use of a single thread through non-blocking, which makes the end-product more scalable and reusable.

Unfortunately, due to time constraints, the catchPokemon module was not implemented with Twisted.

# Data storage and access

How the data is stored has a great impact on the speed and ease at which you can access the necessary data .

## Data stored in JSON files and .TXT files

### JSON format

**Pokedex :**  The pokedex is a dictionary(dict1) which contains the pokemon ID as a key and a dictionary(dict2) as a value. The dictionary(dict2) stores the info of the pokemon corresponding with that ID; the key is the stat name and the value is the stat value. The stat value is either a string, integer, list or dictionary. Please refer to data/pokedexv3.json for greater clarity.

**Player accounts(server-side):** is a dictionary where the username of the player is the key; the corresponding value is a dictionary(dict2) that stores info associated with this account. In the current design, the dictionary(dict2) stores the player’s access\_toke and password. Please refer to data/players.json for greater clarity.

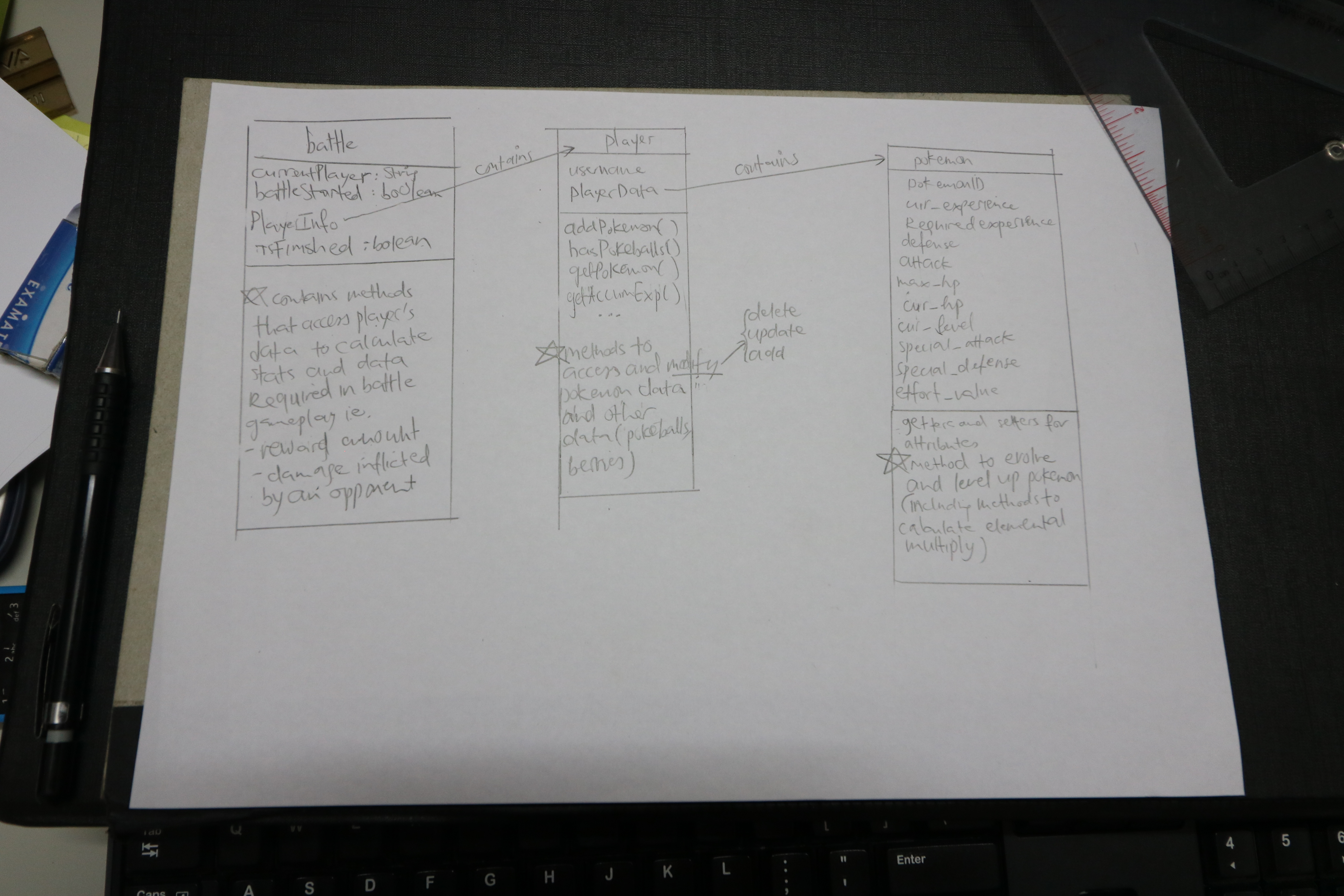
**Player data(server-side):** The name of the json file is the player’s username. The pokemon owned by a player have one of two statuses : “alive”, “fainted” based on their hit points(hp). Thus the pokemon are stored in two keys : “alive” and “fainted”. Information about the quantity of other items (pokeballs and berries collected while in the catchPokemon module) is also stored in the json file. Please refer to data/playerData folder for greater clarity.

### TXT file

**Pokestops(server-side):** The coordinates of the pokestops are fixed and they are stored in a text file, with each line containing two integers, the first is the x coordinate and the second is the y coordinate. Please refer to data/pokestopsX.txt for greater clarity. X refers to the size of the grid (X x X grid) that the pokestops were generated for.

**Access token(client-side):** The client stores and access\_token they received from a client; this access\_token allows then sign in through open authentication. The text file stores the username and the corresponding access token; this data will be read from the text file and sent to the server when the validationC.py is run. If the file is not found, the user have to sign in with their username and password. Please refer to player1/data/access\_token.txt for greater clarity.

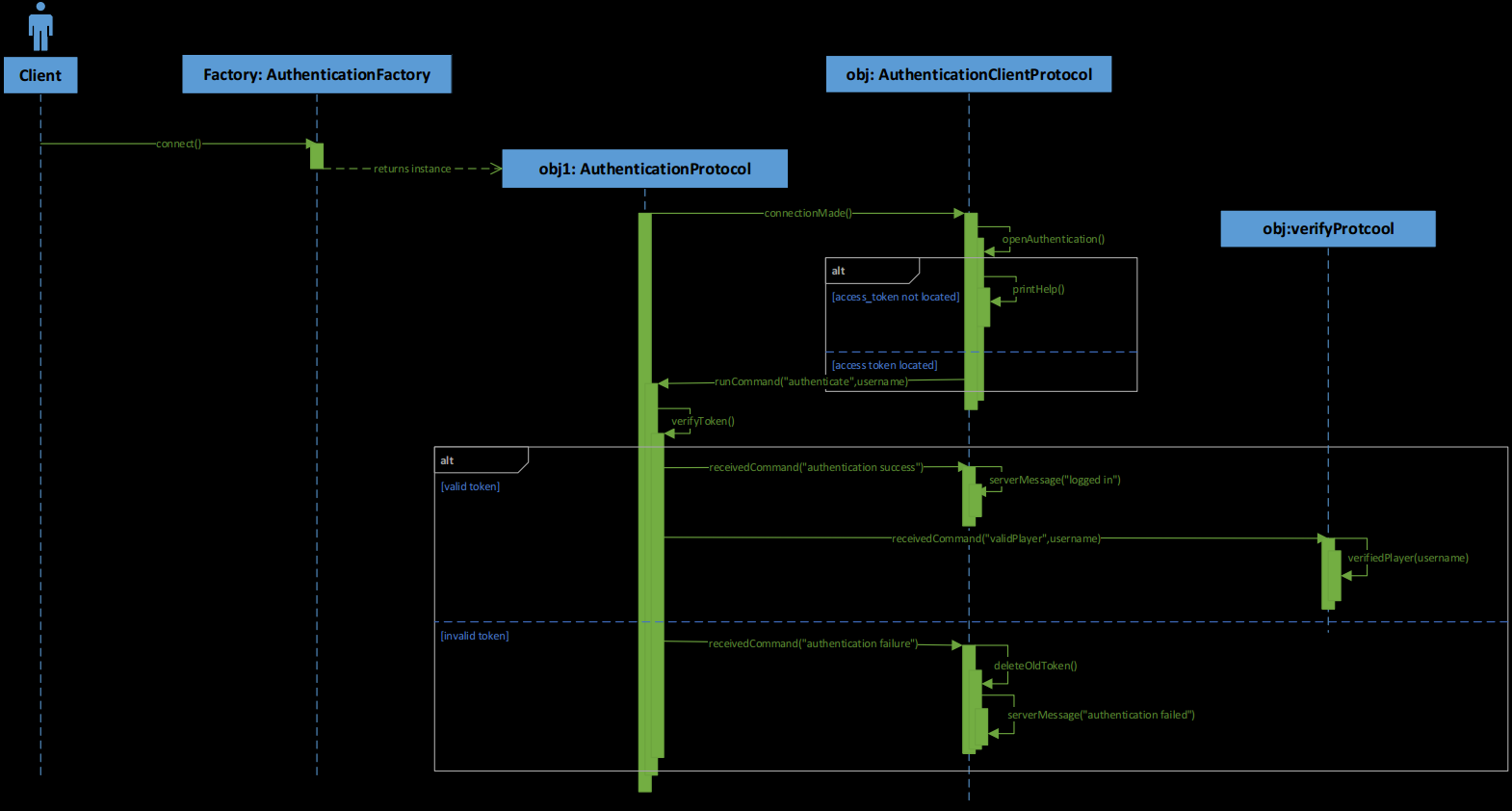
## Data in classes



# Sequence diagram

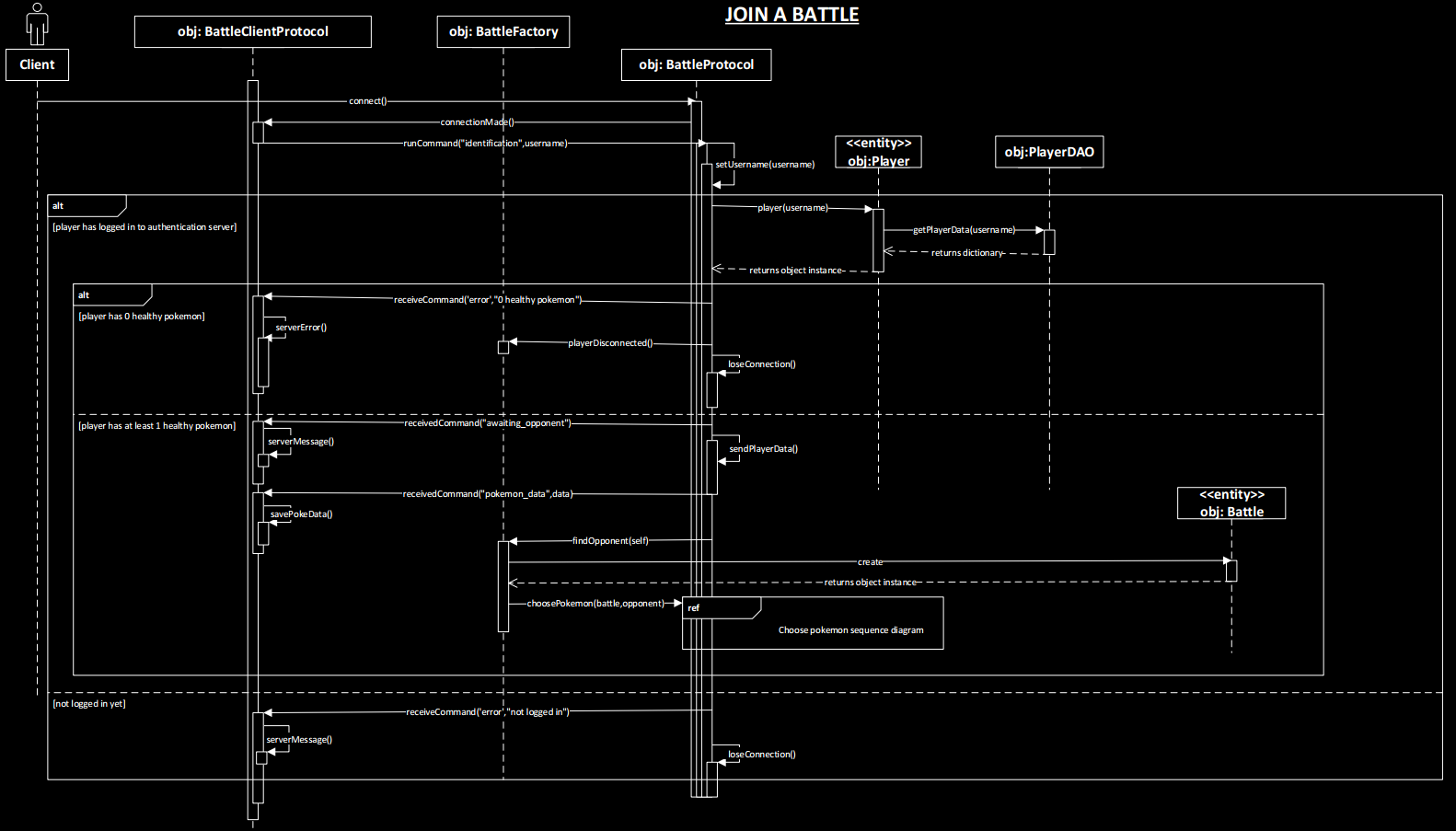
## 2.1. Authentication Module

Open authentication with access token



## Battle Module

Join a battle



Make an attack(server side processing)

Refer to the pdf: sequence diagrams.pdf

## Catch Pokemon Module

Catch pokemon,pokeballs,berries

# 