**VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY**

**INTERNATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**


# NET-CENTRIC PROGRAMMING

# IT096IU

# [Final Project: Pokémon]

**Submitted by**

Tran Thien ITITIU21318

Le Hoang Vi ITITIU21343

Lab Section: Presentation Session

Course Instructor: Dr. Le Thanh Son

# GRADING GUIDELINE FOR LAB REPORT

| Number | Content | | Score | Comment |
|---|---|---|---|---|
| 1 | **Format (max 9%)** | | | |
| | - Font type | Yes No | | |
| | - Font size | Yes No | | |
| | - Lab title | Yes No | | |
| | - Page number | Yes No | | |
| | - Table of contents | Yes No | | |
| | - Header/Footer | Yes No | | |
| | - List of figures (if exists) | Yes No | | |
| | - List of tables (if exists) | Yes No | | |
| | - Lab report structure | Yes No | | |
| 2 | **English Grammar and Spelling (max 6%)** | | | |
| | - Grammar | Yes No | | |
| | - Spelling | Yes No | | |
| 3 | **Data and Result Analysis (max 85%)** | | | |
| **Total Score** | | | | |

Signature:

Date:

# Table of Contents

## Nomenclature

TCP: Transmission Control Protocol

UDP: User datagram Protocol

DNS: Domain Name System

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

BGP: Boarder gateway Protocol

NAT: Network Address translation

# Chapter 1: Introduction

## 1. Abstract

Game net-centric, also known as network-centric gaming, represents a paradigm shift in the design, development, and deployment of video games. This approach leverages the extensive capabilities of modern networking technologies to enhance gameplay experiences, streamline development processes, and foster more vibrant gaming communities. At its core, game net-centric focuses on the seamless integration of multiplayer functionalities, cloud-based services, and real-time data analytics to create dynamic and interactive gaming environments.

The key features of game net-centric include robust multiplayer frameworks, which allow players to interact in complex, persistent virtual worlds. These frameworks are supported by scalable cloud infrastructure that ensures consistent performance and accessibility. Additionally, real-time data analytics enable developers to monitor player behavior, optimize game balance, and deliver personalized content, thus improving player retention and satisfaction.

Furthermore, game net-centric emphasizes interoperability and modularity, facilitating the integration of diverse systems and services. This is particularly important for cross-platform play, where gamers can participate regardless of their hardware, be it a console, PC, or mobile device. The use of advanced networking protocols and security measures ensures that the integrity and confidentiality of player data are maintained, which is crucial in the context of online interactions.

In conclusion, game net-centric represents a transformative approach in the gaming industry, harnessing the power of networked systems to create richer, more engaging, and more adaptive gaming experiences. As technology continues to evolve, the principles of game net-centric are likely to drive further innovations, shaping the future of interactive entertainment.

## 2. Objective:

This Pokémon project will concentrate on learning the basics of how internetworking applications are built and designed. This will be the foundation for moving into more advanced application programming, which will discuss several programming topics support for developing network applications: TCP/IP, stream, threads, socket, client/server…

To allow students to gain practical experience of the design, prototyping, testing, and evaluation stages of network application development.

To allow students to gain practical experience of using the TCP and UDP communications protocols at the programming level.

To strengthen the student's understanding of networking and communication issues by relating theory and practice.

## 3. The technique and tools use:

Programming tools using for this project: Golang (Go)

IDE used: Visual Studio Code.

Discussions: Zalo, Msteam

Resposibility: Github

## 4. Overview of Pokemon project:

Pokémon is a series of video games developed by Game Freak and Creatures Inc. and published by Nintendo as part of the Pokémon media franchise. First released in 1996 in Japan for the Game Boy, the main series of role-playing video games (RPGs) has continued on each generation of Nintendo's handhelds. Games are commonly released in pairs—each with slight variations—and then an enhanced remake of the games is released a few years after the original releases. While the main series consists of role-playing games, spinoffs encompass other genres, such as action role playing, puzzle, and digital pet games. As of February 2016, more than 279 million units have been sold worldwide, more than 200 millions of which from the main series, making it the second best-selling video game franchise, behind only Nintendo's own Mario franchise. The franchise's mascot is Pikachu

# Chapter 2: Contribution

| Name | ID Student | Responsibility | Contribution |
|---|---|---|---|
| Trần Thiện | ITITIU21318 | Build Game Logic and Manager | 50% |
| Lê Hoàng Vĩ | ITITIU21343 | Crawl Database and System | 50% |

# Chapter 3: Game rules and feature

**Firstly**, The Pokémon battle allows two people to attend via network. Each player will pick 3 Pokémon's from their Pokémon list to join a battle.

Secondly, the first Pokémon of 2 players who has higher speed will take first move.

Notice**:** Player can switch Pokémon during battle, turn will end after switching. Player must switch Pokémon if the active Pokémon got killed in battle.

**Secondly**, A Pokémon attack will be randomly chosen between normal attack and special attack A special attack will have all elemental effects of a Pokémon. Highest elemental damage will be applied to target Pokémon

**Next**, when one player announces the surrender situation in a battle, in automatically, the opponent will win.

**Finally**, each Pokémon of winning player will get 1/3 total of accumulated exp of all Pokémons on the losing team.

## ❖ General feature:

➢ **Player Management**

- Players can join the game with a unique username.
- Players can leave the game.
- Players can send public messages to all players or private messages to specific players.

- **Pokémon Management**
  - Players can view a list of available Pokémon.
  - Players can choose three Pokémon for a battle.
- **Battle Management**
  - Players can request a battle with another player.
  - Players can accept or deny battle requests.
  - Players can pick three Pokémon for battle.
  - Players can attack the opponent's Pokémon.
  - Players can change the active Pokémon.
  - Players can surrender a battle.

## ➢ Detailed Features:

**The detailed Features consist of several Player Commands:**

**@pokedex pokemonID/pokemonName**

- Get information of the particular Pokémon by ID or Name

**@join username**

- Join the game with the specified username
- Validates if the username is unique and not already taken

**@all message**

- Send a public message to all players in the game

**@private receiver message**

- Send a private message to the specified player

**@quit**

- To quit the game.

**@battle opponent**

- Send a battle request to the specified player

- Validates if the player is not already in a battle

**@accept sender**

- Accept a battle request from the specified player
- Initializes the battle state.

**@deny sender**

- Deny a battle request from the specified player.

**@pick pokemonID**

- Pick the Pokémon for the battle by type the id

**@attack**

- Attack the opponent's active Pokémon
- Handles turn-based attacks

**@change**

- Change the active Pokémon during a battle

**@change**

# Chapter 4: Game Structure

## 1. General Purpose:

The primary objective in designing the sophisticated structure of the Pokedex Game application is to create a digital adaptation that faithfully captures the essence of exploring and cataloging Pokémon species. This structure aims to establish a robust foundation for an immersive, user-friendly, and adaptable gaming experience, leveraging the capabilities of digital platforms.

The goal is to provide players with an engaging journey through the world of Pokémon, where they can discover, collect, and learn about various Pokémon species akin to how trainers explore in the Pokémon universe. The application will feature an intuitive interface for browsing and managing Pokémon entries, detailed information on each species, interactive features for

tracking catches and achievements, and potentially multiplayer elements for trading and battling Pokémon with other players.

By emphasizing fidelity to the Pokémon universe and optimizing for user engagement and accessibility, the Pokedex Game aims to be a comprehensive digital companion that enhances the experience of Pokémon enthusiasts and players alike.

## 2. General Game Structure:

In our Pokémon game built using Golang, the `main.go` file is responsible for managing the game server, ensuring smooth operation and handling of various game mechanics. The client-side interactions are handled by the `client.go` file, which processes and responds to messages from players, facilitating seamless communication within the game. To populate the game's database with comprehensive Pokémon information, `crawler.go` is utilized to scrape data from the Pokémon Database website. This scraped data is then stored in `pokedex.json`, providing a rich resource for the game to reference all Pokémon attributes and details. Lastly, the game maintains `playersPokemon.json`, a file that keeps track of players and their owned Pokémon, ensuring that each player's collection is accurately recorded and updated as they progress through the game.

❖ **Main.go:**

The Pokémon game described by the code follows a client-server architecture using the UDP protocol. Here’s a breakdown of the general game structure:

➢ Initialization and Setup

Constants and Data Files:

- HOST: "localhost"
- PORT: "8080"
- TYPE: "udp"
- pokedexData: "src\\pokedex.json" (Pokedex data)
- playerpokemonsData: "src\\playersPokemon.json" (Player's Pokémon data)

Global Variables:

- pokedex: Array to store all Pokémon data.
- playersPokemons: Array to store all players' Pokémon.
- players: Map to store currently online players.
- inBattleWith: Map to track players currently in battle.
- gameStates: Map to manage ongoing battles.

➢ Data Structures

Pokemon:

- Basic attributes like ID, Name, URL, and detailed information (PokeInfo).

PokeInfo:

- Attributes like Types, HP, ATK, DEF, etc.
- Type defenses against various Pokémon types.

Player:

- Player's information including name, address, Pokémon holdings, battle requests, active Pokémon, and battle ID.

PlayerPokemon:

- Data structure to store a player's Pokémon.

PlayerPokeInfo:

- Individual Pokémon details like ID, Name, Level, Stats, and Type defenses.

BattlePokemon:

- Structure for Pokémon involved in battles with relevant stats.

Battle:

- Manages ongoing battles with details like battle ID, players involved, active and beating Pokémon, turn management, and status.

- Core Functionalities

Data Loading:

- loadPlayerPokemon(filename string): Loads players' Pokémon data from a JSON file.
- loadPokedex(filename string): Loads the Pokedex data from a JSON file.

Main Function:

- Initializes the server, loads necessary data, and listens for incoming UDP messages.
- Handles incoming messages and spawns goroutines to process commands.

Command Handling:

- handleMessage(message string, addr *net.UDPAddr, conn *net.UDPConn): Processes commands received from clients.
- Supports various commands (@join, @all, @quit, @private, @battle, @accept, @deny, @list, @pokedex, etc.).
- Manages player interactions, battle requests, and Pokémon battles.

Battle Management:

- Handles initiating and conducting battles, including selecting Pokémon, attacking, and managing turn-based actions.
- Uses gameStates to keep track of battle states and manage transitions from one state to another.

- Player Interactions
  - Players can join the game, initiate battles, send messages (both public and private), and list their Pokémon.
  - Battles are initiated via the @battle command and accepted or denied with @accept and @deny.
  - During battles, players can select Pokémon, attack, or change active Pokémon.

- Game States
  - Battles have states like "waiting", "inviting", and "active" to manage the flow of the game.
  - Each battle track which Pokémon are currently active, which player's turn it is, and the status of each player's Pokémon.

❖ **Client.go:**

The game is a multiplayer text-based Pokémon battle simulation where players connect to a server using UDP (User Datagram Protocol) to join battles, pick Pokémon, and engage in turn-based combat. Players interact with the game using simple text commands and receive responses from the server to guide their actions.

- Connection Setup:
  - The game resolves a UDP address (localhost:8080) and establishes a connection to the server.
  - It prompts the player to enter a username and attempts to join the game by sending a join request to the server.
- Username Validation:
  - The server checks if the username is unique. If not, it prompts the player to choose a different username.
  - Once a unique username is accepted, the player successfully joins the game.
- Message Handling:
  - A separate goroutine (receiveMessages) is responsible for receiving and processing messages from the server.
  - Messages include various game prompts and updates such as Pokémon list requests, battle status updates, and action confirmations.

- User Interaction:
  - The main loop allows the player to input commands and send them to the server.
  - Commands include joining battles, picking Pokémon, attacking, and changing Pokémon during battles.
  - Specific commands are required to handle different game actions (e.g., @pick, @attack, @change).
- Game Logic:
  - Players can only attack if they have an active Pokémon that is not required to be changed.
  - The game handles various states like waiting for battle, selecting Pokémon, battling, and checking the results of battles.
- Battle Mechanics:
  - Players pick three Pokémon to use in battles.
  - Battles are turn-based, with each player's Pokémon taking turns to attack or perform actions.
  - The server notifies players about the status of their Pokémon, such as successful attacks, Pokémon changes, and battle outcomes.
- Synchronization and Mutex:
  - Mutex (mu) is used to handle synchronization, ensuring that shared resources (like maps for tracking battle states) are accessed safely.
- Server Responses:
  - The game listens for various server responses that guide the player through the game. These include:
  - Battle Commands: Initiating and progressing through battles (@accepted_battle, @pokemon_start_battle).
  - Pokémon Management: Listing then picking Pokémon (@list_then_pick_pokemon, @pokemon_picked).
  - Battle Outcomes: Notifying win/loss (@win, @lose).

- Turn Management: Handling turns during battles (@opponent_attacked, @you_attacked).

❖ **Crawler.go:**

The application is a Pokémon data extraction tool that scrapes Pokémon details from a specific website and stores the data in a structured JSON format. It fetches information such as Pokémon IDs, names, types, and detailed stats including HP, attack, defense, and type defenses. This data is then saved into a JSON file called pokedex.json.

➢ Main Components:

1) HTTP Request and Response Handling:

- The application begins by making an HTTP GET request to the Pokémon database website (https://pokemondb.net/pokedex/national).
- It reads and parses the HTML response from the website.

2) HTML Parsing:

- The HTML response is parsed to extract Pokémon information.
- The getPokedex function traverses the HTML document to gather basic information about each Pokémon, such as ID, name, types, and a link to more detailed information.

3) Data Structuring:

- The data is structured into custom Go structs (Pokedex, PokeInfo, TypeDef, and ExpStats) to facilitate easy manipulation and storage.

4) Detailed Information Extraction:

- For each Pokémon, a detailed extraction function (getDetail) is called using the link gathered previously.

- This function fetches more specific data such as individual stats (HP, attack, defense, etc.) and type defenses.

5) JSON Serialization:

- The collected and structured data is serialized into JSON format using Go's encoding/json package.
- The JSON data is indented for readability and written to a file named pokedex.json.

6) Error Handling:

- The application includes error handling for HTTP requests, reading responses, and file operations to ensure robustness.
- Errors are logged to the console to inform the user of any issues encountered during the execution.

➢ Detailed Functions:

1) main Function:

Manages the overall flow of the program: connecting to the website, fetching the HTML content, parsing the content, and writing the output to a file.

2) getPokedex Function:

- Recursively traverses the HTML nodes to collect basic Pokémon data.
- Extracts and organizes information like ID, name, types, and links for further details.

3) getDetail Function:

- Fetches and parses detailed information for each Pokémon from the individual detail pages.
- Collects stats and type defense information and stores them in the PokeInfo struct.

4) Helper Functions:

    - Several helper functions (getInsideTag, getOnce, getStringElement, getStatNumber, and getRatioDef) are used to extract specific pieces of information from the HTML nodes.

➢ Overall Flow:

1) The program starts by connecting to the Pokémon database website and fetching the HTML content.
2) It parses the HTML to gather basic Pokémon information and then fetches detailed information for each Pokémon.
3) All the collected data is structured into Go structs and serialized into a JSON file.
4) The final JSON file, pokedex.json, contains a comprehensive list of Pokémon with their detailed stats and type defenses, making it a useful resource for further analysis or usage in other applications.

```
{
  "ID": "#0001",
  "Name": "Bulbasaur",
  "types": [
    "Grass",
    "Poison"
  ],
  "URL": "/pokedex/bulbasaur",
  "Poke-Information": {
    "HP": 45,
    "ATK": 49,
    "DEF": 49,
    "Sp.Atk": 65,
    "Sp.Def": 65,
    "Speed": 45,
    "Type-Defenses": {
      "Normal": 1,
      "Fire": 2,
      "Water": 0.5,
      "Electric": 0.5,
      "Grass": 0.25,
      "Ice": 2,
      "Fighting": 0.5,
      "Poison": 1,
      "Ground": 1,
      "Flying": 2,
      "Psychic": 2,
      "Bug": 1,
      "Rock": 1,
      "Ghost": 1,
      "Dragon": 1,
      "Dark": 1,
      "Steel": 1,
      "Fairy": 0.5
    }
  }
},
```


Figure 1: An Pokemon Data in Pokedex.json

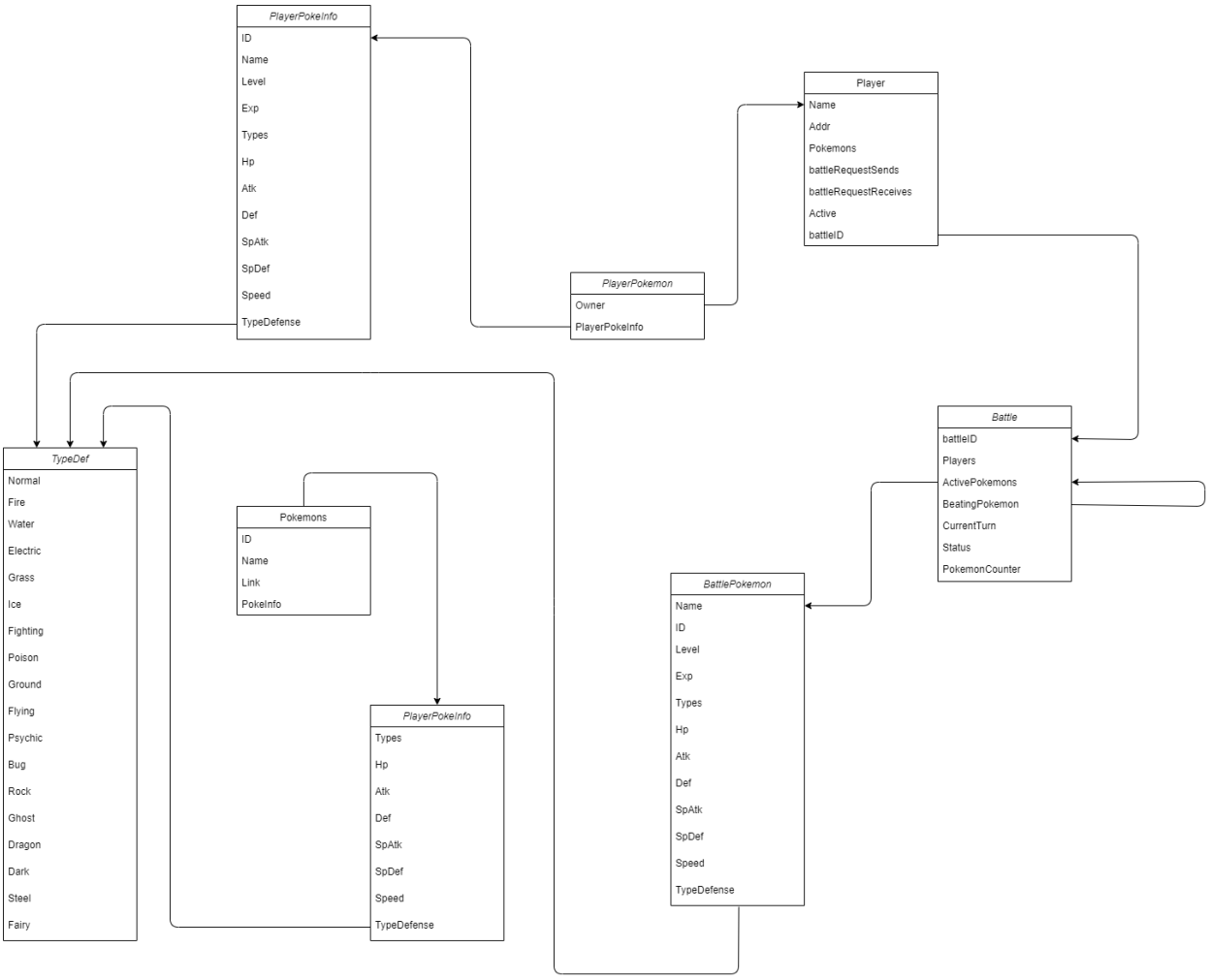


Figure 2: Class Diagram

```
1   List of game instructions:
2
3   To chat all:                    @all message
4   To chat private:                @private receiver message
5   To request a battle:            @battle opponent
6   To accept a battle:             @accept sender
7   To deny a battle:               @deny sender
8   To pick pokemons:               @pick pokemonID(in your owned pokemon list)
9   To change pokemon:              @change pokemonID(in your owned pokemon list)
10  To acttack:                     @attack
11  To find pokemon info            @pokedex pokemonID
12  To change pokemon in the battle @change
```

Figure 3: Game Instruction

# Chapter 6: Conclusion

In this game, players manage a diverse roster of Pokémon, each with unique attributes and abilities, creating a rich and dynamic gameplay experience. The variety in Pokémon teams, with differing levels, experience points, and specializations, fosters strategic depth and personalization for each player. While certain Pokémon like Pikachu are popular choices across multiple players, the inclusion of distinct Pokémon for each player ensures that no two teams are exactly alike. This variety not only enhances the competitive aspect of the game but also keeps the gameplay engaging and unpredictable. By carefully managing their Pokémon's growth and leveraging their strengths, players can develop unique strategies to outmaneuver their opponents, making the game both challenging and rewarding. Overall, the game's structure provides a well-balanced platform for both novice and experienced players to enjoy and master.

# Chapter 7: Reference

1. Nintendo. (1996). Pokémon Red and Blue [Video game]. Nintendo.
2. Pokémon. (n.d.). *The Official Pokémon Website*. Retrieved June 13, 2024, from https://www.pokemon.com
3. GolangDocs. (n.d.). *Goroutines in Golang*. Retrieved June 13, 2024, from https://golangdocs.com/goroutines-in-golang
4. W3Schools. (n.d.). *Go Tutorial*. Retrieved June 13, 2024, from https://www.w3schools.com/go/index.php
5. freeCodeCamp. (n.d.). *TCP and UDP Protocols*. Retrieved June 13, 2024, from https://www.freecodecamp.org/news/tcp-and-udp-protocols/
6. (n.d.). *List of Pokémon*. Pokémon Database. https://pokemondb.net/pokedex/national