

CIS-427 Assignment 1

Students

- Senan Aboluhom: senan@umich.edu
- Mohammed Alkhyat: malkhyat@umich.edu

Project Overview

- Platform: Linux (on UMD servers)
- Programming Language: C++
- Database: SQLite3
- Port Number: 4869

Compilation Instructions

- Need g++ compiler
- Need sqlite3.c and sqlite3.h files in the project directory
- Move into the directory: `cd pokemon_trading`
- To compile the server and client we can run `make all` to compile both
- To start the server run the following command: `./pokemon_server`
 - Will open or create the `pokemon_store.db`
 - Will create `Users` and `Pokemon_cards` table if they don't already exist
 - Creates 4 users: mary, john, moe, and root with a balance of \$100
 - Listens for the client on port 4869
- To start the client run the following command: `./pokemon_client`
 - If the hostname isn't provided, the client defaults to `localhost(127.0.0.1)`
- Can connect up to 10 users at the same time

Implemented Commands

- All the commands from version one of the project are still implemented so we will not go over them again over here since they have already been documented. Will just go over new implementations
- LOGIN: Log into the system
 - Format: `LOGIN<username><password>`

-example: LOGIN mary Mary01

-
- LOGOUT: Log out of the system
 - Format: LOGOUT
 - Example: LOGOUT
- DEPOSIT: Add money to your account
 - Format: DEPOSIT <amount>
 - Example: DEPOSIT 50.00
- LIST: View your card collection. Has been modified so if the user uses this command, it will display all cards that all users have
 - Format: LIST
 - Example: LIST
- WHO: List all active users. Only the root can use this command
 - Format: WHO
 - Example: WHO
- LOOKUP: Search for any cards by name. The name can be a full or partial match
 - Format: LOOKUP <card_name>
 - Example: LOOKUP Char
- SHUTDOWN: Shut down the server. Only the root can use this command
 - Format: SHUTDOWN
 - Example: SHUTDOWN

Student Contributions

Senan Aboluhom:

- Implemented the LOGIN command
- Implemented the DEPOSIT command
- Implemented the WHO command
- Implemented error handling for commands implemented
- Removed the ID requirement for the commands implemented since it wasn't necessary

Mohammed Alkhyat:

- Implemented the LOGOUT command
- Modified the LIST command to allow the root to view all cards
- Implemented the LOOKUP command

- Implemented the SHUTDOWN command
- Implemented error handling for commands implemented
- Removed the ID requirement for the commands implemented since it wasn't necessary

NOTE: We both met up together and worked on the thread implementation, active users, thread pool, select(), etc. together because we found it tricky to split this up and combine it later, so we decided to work on this part together.

Error Handling

The server provides error handling for the following cases:

- 400 invalid command – Unrecognized command sent to server
- 403 message format error – Incorrect command format or missing parameters
- 403 Insufficient Funds – User does not have enough money to complete purchase
- 403 Insufficient Cards – User does not have enough cards to sell requested quantity
- 403 Card not found – User does not own the specified card type
- 404 User Not Found – User ID does not exist in database
- 500 Database Error – Database operation failed
- 503 Server capacity – Limit exceeded for server capacity

Bugs

As of the second implementation of this project, we did multiple tests off screen and haven't been able to find any bugs or ways to crash our program as of our knowledge.

Demo Video

A video demonstration of all commands and error handling:

<https://youtu.be/Y640xIvn12o>

