Dice Poker Project Documentation

Kamil Lesiński, Adam Sobczuk, Jan Ludwicki

This project is a simple server-client implementation of the Yahtzee game using Python and sockets for communication. We take our inspiration from the first game of the Witcher Series (https://wiedzmin.fandom.com/wiki/Ko%C5%9Bciany_poker) and popular yahtzee game (https://en.wikipedia.org/wiki/Yahtzee). The server manages the game state and communicates with the clients, which are the players of the game.

Files

server.py: This file contains the server logic. It handles client connections, processes client commands, and manages the game state.
client.py: This file contains the client logic. It connects to the server, sends commands, and receives responses.
yahtzeeGame.py: This file contains the YahtzeeGame class, which encapsulates the game logic.
start.bat: This file contains bash commands for running three separate instances to handle communication (one server and two clients)

How to Run

Start the application by running ./start.bat in a terminal in project directory
OR
Open three separate command line windows
Run python server.py in one and python client.py in others

Commands

Players can enter the following commands:
roll: Rolls the dice. If it's not the player's turn, the server will respond with "It's not your turn".
keep <indices>: Keeps the dice at the specified indices and rolls the rest.
score <category>: Calculates the score for the specified category.
add <category>: Adds the score for the specified category to the player's total score.
reset: Resets the game.
exit: Disconnects the client from the server.

Game Logic

The game logic is implemented in the YahtzeeGame class. It manages the dice, the players' turns, and the players' scores. It also provides methods for rolling the dice, calculating the score, and adding the score.

Communication

The server and clients communicate using sockets. The clients send commands as strings, and the server responds with the result of the command. The communication is synchronous: the client waits for a response after sending a command.

Limitations

The server can handle multiple clients, but the game logic is designed for two players.
The server does not handle disconnections gracefully. If a client disconnects, the server will continue to wait for its turn.
The server does not validate the commands. If a client sends an invalid command, the server will respond with "Unknown command".