Riddleet App: CENG421 Project Report

Burak Keser - 260201057

Riddle game with group chat made with Python **sockets** and **curses**. Client and Server are tested on **Python 3.9** so you need to execute them on *Python 3.9*+. I only used standard python libraries.

Each player can open or join a room. Game gets started by the owner of the group. Each question/riddle is 10 points.

```
Execution:
    client: python3 client.py
    server: python3 server.py
```

Riddleetclient

> Uses curses, works on terminal. Includes a client server and UI element for consoles.

User Commands

```
/q                  quit the game,
/r <''|group|server> clean the view,
/n <name>:          set your name!,
/o                  open a room!,
/k <playerID> :     Kick a player!,
/j <id>:            joins to a playroom!,
/d                  leave the room!,
/s                  Start the game!,
/p                  Print the players/leaderboard!,
/!<msg>             send a message to your room!
```

Commands with ! at the end of their explanation needs server connection to execute. Each online command sends a request to the one of server endpoints. Client does not hold and data on its own and mirrors the state of the player on the server.

Connection

Client server, reacts to data sent from the server and display the changes and requests on the IU. Mounted on its own thread.

Endpoints for server to use for **requests/responds**.

Route	Description
Information	

Route	Description
set name <name>	Sets the name on client.
set id <ID>	Sets the id on client.
Room	
open room	Response for opening a room, sets the current room.
join room <roomID>	Response for joining a room, sets the current room.
leave room	Response for leaving a room, sets the current room null. Leaving from ongoing game clears question and game information headers.
Message	
send notify <notification>	Notification endpoint for server to sends notifications about group.
send message <message>	Message endpoint for server to sends messages from
Game	
start game	Response affirming the game has been started, sets the
end game	Endpoint for server to stop client send more messages as an answer.
set question <question>	Endpoint for server to set current question.
set score <score>	Endpoint for server to set current score of the player.
accept answer <score>	Accept response to players answer, puts a **"✓"** mark to the Answer element on the Header and sets the current score.
reject answer	Reject response to players answer, puts a **"X"** mark to the Answer element on the Header.
Player List	
print lb <players>	Response/Endpoint for user/server to print out the list of users in the room with score(leader board) or not (user list) according to the state of the room. (In game: leader board, Waiting: user list, without scores).

Display

Contains UI implementation, uses components. Contains Boxes for.

Component	Description

Component	Description
Header	Contains server, player and question information.
Prompt	Handles user input.
UserContext	User prompts/commands.
GroupContext	Group activity, messaging, group related commands.
ServerContext	Server responses.

Components

Contains implementation details for **Header**, **Prompt** and **Context**

Timer

A mock timer component. Manipulates the time element on the header. Somewhat tries to match with the server side but does not have a connection to it. Manually set to count from 15 but can be tweaked to count from specified numbers if needed.

Riddleetserver

> Supervise the game and chat. Has a similar arrangement to RESTful API, gets requests, react to requests and returns a response if needed. Instead of database uses dictionaries and does not store the data. After every reset server starts clean. There is no sign up, every socket connected to server starts playing.

Room Structure

```
Rooms
    roomID
        owner: str
        status: WAIT | STARTED
        maxSize: int
        currentSize: int
        game: Game
        players:
            ID:socket
        playerNames:
            ID:str
        playerScores:
            ID:int
        playerAnswered:
            ID:bool
```

Server can access the rooms by the roomID's stored in each player thread after joining or creating a room. Each Room can only have one owner. Room has status indicating if the game is ongoing or

room is waiting for players. Room has dynamic maxSize but for now changing maxSize did not implemented.

Game loop started by the owner of the group and reference to its object is given in the dictionary for answering functionality. player sockets, names, scores and status indicating if player answered the current question is also stored in the room.

Player Thread

Endpoints for client/players to use for **requests**.

Route	Description
set name <name>	Sets the player name on the server and sends a set name response to the client to match the names.
open room	Opens a room in the server and sends open room response to the client which sets the roomID on client to the joined room.
join room <roomID>	Joins a room with the id "roomID" if it exist.Responds with appropriate errors or if successful sends join room response which sets the roomID on client to the joined room.
leave room	Leaves the current room and sends leave room response to the client which sets the room to null. If owner leaves, selects a new random owner.
send message <message>	Echos the sent message to all users in the room.
start game	Totally starts the game.
send answer <answer>	Answer endpoint, checks with current question answer and if correct increases the score and returns accept answer response, if answer does not match return reject answer response. If encounter error returns an error to client.
get players	Get the players in to room, used for printing leaderboard/player list

Game

Contains the main game loop, started by the **start game** request by the ***owner*** of the room. Currently each question has 15 seconds and each game session has 5 question but these can be connected to endpoints easily. Loops through the questions, sends them to each player in the room and waits for answers for 15 seconds and after that changes the question so on so forth. With each iteration and at the end of the game, sends request to client to print leaderboard.

Question

For question utils, class for question storing and comparison and function to read riddles/questions from riddles.csv.

For execution screenshots you can look client.gif and server.gif or screenshots folder.