Analysis:

The game is grouping 10 (configurable) players together in real time manner (grouping time is 30 seconds). Player contains information about the country/location, so the players are distributed all over locations and might be globally everywhere.

As this is a real time game, I think designing a distributed system all over the world to support the game experience is sensible, for example, players in Finland will be served by Finnish backend server. This will improve the latency for the end user, therefore, improve the game experience.

I play League of Legends, also a game matching of 10 players together into a match based on the player rank. So, I believe that it is sensible that I would assume that the game is quite complex and there would be a game engine for it. I think we can go with one of 2 options

- HTTP request between server and client, client needs to call the service for update, the server can remain stateless and scale easily without any other complexity. The downside is that the client might do a lot of requests, and with many clients, the server might be bombarded, and the system might do DDoS itself when the system grows.
- Web socket between server and client. The game running on client device can run and receive the data passively, by the server will push the data to clients through the socket. Server need to be stateful to store the active connection to the client. The good side is that in the game, I think a lot of game events, data needs to be broadcast, and this is quite simple when we have a stateful service which holds the sockets, and the sync between player is also better, as the events of the game will be received by the players at the same time, instead of depends on the client code. The downside of this one that it introduces the complexity of the distributed system, where the deciding the server that the player would connect to and how to handle when the backend service fail is more complicated.

I think the first approach is straightforward and quite easy, where the player experience might become an issue when the game evolves. And to show case my experience, I would go for the second option, which I believe I could provide better player experience (as I am a gamer too).

The infrastructure and the backend service of the system should be configurable and self-organized. I have some knowledge how Kafka brokers do their self-organized method. Anyway, I would not choose the leader-follower pattern, but they are all leader, and they can promote each other to serve the clients.

The server should be configurable like where an instance can have id and(or) name, and an address that the client could connect to, such as “fi01.futureplaygame.com”, where it can be discovered by the client.

Multiple instances can be instanced by the game code, but different configurations, I would take example in my code with

“fi.futureplaygame.com”,

“se.futureplaygame.com”

(Where the Finnish server is better than Sweden server, probably as Finnish players are happiest players in the world 😊)

The scaling within 1 region/country could also be done, but I will not going to detail on that as it is just configuration matter.

The game I also assume: the game is full of the players with players from different levels range, therefore, the game is usually crowed and the criteria for match making usually can find enough players, BUT of course there might be when the late night time, where there are not many active players, so I would only implement a simple algorithm to handle this case and merging the players when needed.

Score Matching:

A formal language how to determine the matching scores of the player is also introduce for flexibility. For example, I will describe a formula that calculate the matching scores with “level / 10”, which means that with the player level, the computation of the score will be player.level /10, and might be rounded. So player level 9 and 11 would end up the same score, which is 1. This value would help determine to group up players group that have the score closely together.

The game logic should also be described by a formal language, for simplicity, I set the base game is that each user will give an input, and then a function defined could be described, for example:

Guess the number: each player guesses 1 number in range 1-100, and then the system will generate a random number, whoever has closest value would win, so the competition description is “RANDOM”

Maximum of points: The client will observe some score extracts from the game, and the server will decide whoever has the best grade will win, and competition descriptions is “MAX”

So, in general, there would be a formal way to describe the game by the game creator, and I will only implement RANDOM and MAX game strategy.

API:

For administrators to create the game

POST /api/v1/game to create the game,

example payload:

{“name":"Guess number","max_player":10,"match_formula":"{\"match\":\"`${country}-pool`\",\"score\":\"level / 10\"}

GET /api/v1/game to get all the games,

POST /api/v1/hint to ask the server for the personal token to join the game, and which server that the client should connect to

Example payload: { "id": "1", "gameId": "1" }

Example response:

{ "token":
"eyJ1c2VyX2lkIjoxLCJ1c2VyX25hbWUiOiJBaW5vIiwidXNlcl9sZXZlbCI6NiwidXNlcl9jb3VudHJ5Ijoi
RkkiLCJtYXRjaF9zY29yZSI6MSwiZ2FtZV9pZCI6IjEifQ==", "service": "localhost:3001/ws" }

Websocket endpoint /ws to serve players during the game.

In the design, I have advertised the connect server for the player, firstly near their location, and then using a match function, to group all of the player having a same name, for example, people located in finland was grouped into the pool called “FI-pool”.

The number of player is configurable through the game administration, by updating/creating the game with max_player selected.

The game could also be scheduled, so the player would wait at most 30 seconds to get into the game, and if the queue is big, there is no need to wait for whole 30 seconds, but the games will be scheduled after it meets the threshold in queue.

I think that the requirement does not require me to implement the simple game to simulate how the game run, but as I never have experienced with WebSocket before as I said in my interview, I think learning it would be great as well, through this opportunity.

Code structure:

I think my organization styles is affected by my experience in coding GO, so that the packages are separated and clear in responsibility.

The implementation is heavily focused on functional programming, as I see the ease of organizing code. I believe that for some people, it is really difficult to read and follow at first, but after a while, I think I see the art in functional code, as I see clean organization, and very easy to test.

The choice of language is JavaScript with NodeJS. I thought that it would be faster to code in JavaScript as generic. I do not really have problems coding with generic-typed language, and I think it is much faster for me not to specify tons of interfaces.

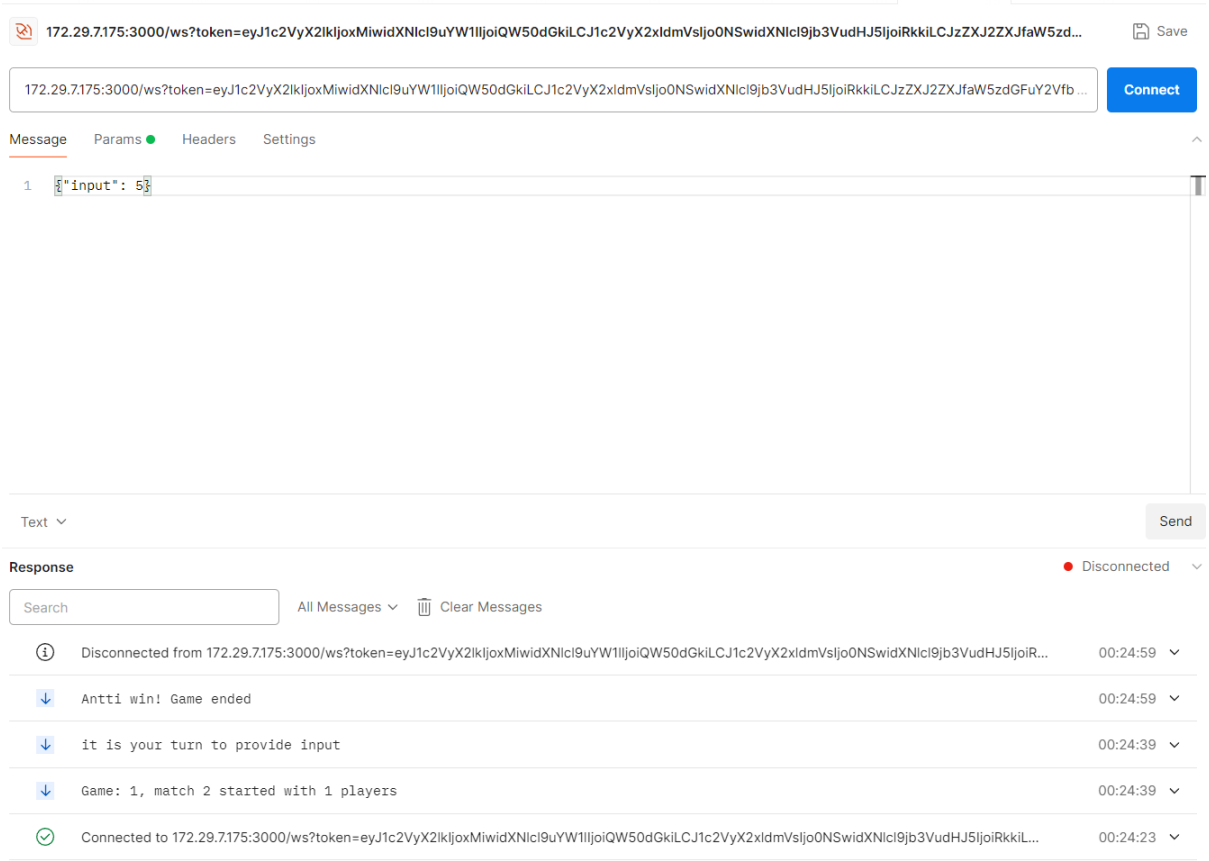
The services are containerized and configurable to be deployed in distributed locations. The way to test it locally is to use “docker compose up”

Obviously, I spend more than 4 hours to complete this, but as I said, I want to learn about the WebSocket and see the meaning of the assignment, to test out my idea how I would implement the game that I have been playing for a decade.

The implementation of server is in folder **/backend,** I have also implemented a simulator for client code that would use the WebSocket to connects to backend services, you can see the output of the game events through the logs of the containers.

Appendix:

A. Interation with the application through websocket



B. Logs from server:

fi-hel-service-1 | Player Aino:1 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Elias:2 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Venla:3 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Matias:4 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Sofia:5 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Olavi:6 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Kerttu:7 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Mikael:8 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Ilona:11 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Antti:12 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Marja:13 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Oskari:14 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Helmi:16 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Liisa:9 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Pekka:15 joined game 1 queue in FI-pool pool

fi-hel-service-1 | Player Juhani:10 joined game 1 queue in FI-pool pool

se-sto-service-1 | Player Astrid:17 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Erik:19 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Karin:20 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Ingrid:18 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Lars:21 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Elin:22 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Johan:23 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Sven:24 joined game 1 queue in SE-pool pool

se-sto-service-1 | Player Anna:25 joined game 1 queue in SE-pool pool

fi-hel-service-1 | Game: 1, match 926529 started with 8 players

fi-hel-service-1 | Game: 1, match 394783 started with 8 players

fi-hel-service-1 | player Aino give input 1

fi-hel-service-1 | player Elias give input 18

fi-hel-service-1 | player Venla give input 14

fi-hel-service-1 | player Matias give input 5

fi-hel-service-1 | player Sofia give input 19

fi-hel-service-1 | player Olavi give input 11

fi-hel-service-1 | player Mikael give input 13

fi-hel-service-1 | player Kerttu give input 15

fi-hel-service-1 | player Liisa give input 7

fi-hel-service-1 | player Juhani give input 16

fi-hel-service-1 | player Marja give input 2

fi-hel-service-1 | player Ilona give input 9

fi-hel-service-1 | player Oskari give input 4

fi-hel-service-1 | player Pekka give input 12

fi-hel-service-1 | player Antti give input 4

fi-hel-service-1 | player Helmi give input 11

se-sto-service-1 | Game: 1, match 3659 started with 9 players

se-sto-service-1 | player Erik give input 6

se-sto-service-1 | player Astrid give input 19

se-sto-service-1 | player Anna give input 12

se-sto-service-1 | player Sven give input 8

se-sto-service-1 | player Johan give input 6

se-sto-service-1 | player Elin give input 4

se-sto-service-1 | player Lars give input 4

se-sto-service-1 | player Karin give input 19

se-sto-service-1 | player Ingrid give input 5

fi-hel-service-1 | Matias win! Game ended

se-sto-service-1 | Astrid, Karin win! Game ended

fi-hel-service-1 | Helmi win! Game ended

fi-hel-service-1 | Player Olavi:6 leave game 1

fi-hel-service-1 | Player Mikael:8 leave game 1

fi-hel-service-1 | Player Kerttu:7 leave game 1

fi-hel-service-1 | Player Matias:4 leave game 1

fi-hel-service-1 | Player Sofia:5 leave game 1

fi-hel-service-1 | Player Venla:3 leave game 1

fi-hel-service-1 | Player Aino:1 leave game 1

fi-hel-service-1 | Player Pekka:15 leave game 1

fi-hel-service-1 | Player Oskari:14 leave game 1

fi-hel-service-1 | Player Helmi:16 leave game 1

fi-hel-service-1 | Player Ilona:11 leave game 1

fi-hel-service-1 | Player Liisa:9 leave game 1

fi-hel-service-1 | Player Juhani:10 leave game 1

fi-hel-service-1 | Player Antti:12 leave game 1

fi-hel-service-1 | Player Marja:13 leave game 1

se-sto-service-1 | Player Astrid:17 leave game 1

se-sto-service-1 | Player Ingrid:18 leave game 1

se-sto-service-1 | Player Lars:21 leave game 1

se-sto-service-1 | Player Erik:19 leave game 1

se-sto-service-1 | Player Karin:20 leave game 1

se-sto-service-1 | Player Sven:24 leave game 1

se-sto-service-1 | Player Johan:23 leave game 1

se-sto-service-1 | Player Elin:22 leave game 1

fi-hel-service-1 | Player Elias:2 leave game 1

se-sto-service-1 | Player Anna:25 leave game 1

C. Logs from simulator

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 926529 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 394783 started with 8 players

it is your turn to provide input

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

it is your turn to provide input

Game: 1, match 3659 started with 9 players

Game: 1, match 3659 started with 9 players

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

it is your turn to provide input

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Matias win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Astrid, Karin win! Game ended

Helmi win! Game ended

Astrid, Karin win! Game ended

Helmi win! Game ended

Helmi win! Game ended

Helmi win! Game ended

Helmi win! Game ended

Helmi win! Game ended

Helmi win! Game ended

Helmi win! Game ended