Game Application – Number Game

Technical Design document

Asif U

April 30th, 2019

**Problem Statement:**

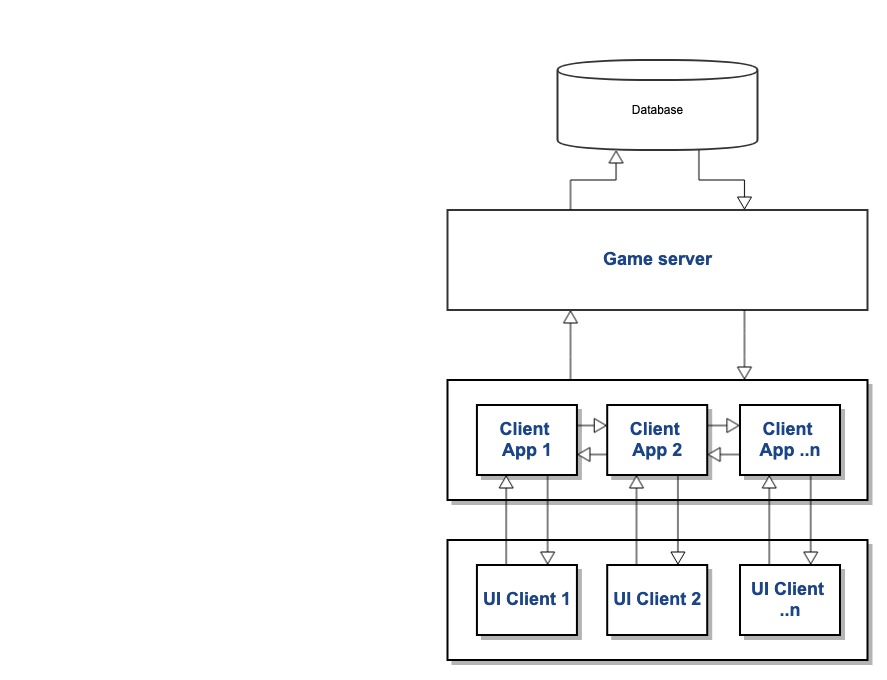
Game Application – Number Game

* Two players at a time can play a game by exchanging of a number where one player will start with a random number then dividing it by 3.
* Both player will exchange the result number till it reaches 1, if the user reaches 1 then he is the winner of the game.
* For each user, separate application should run along with separate UI.

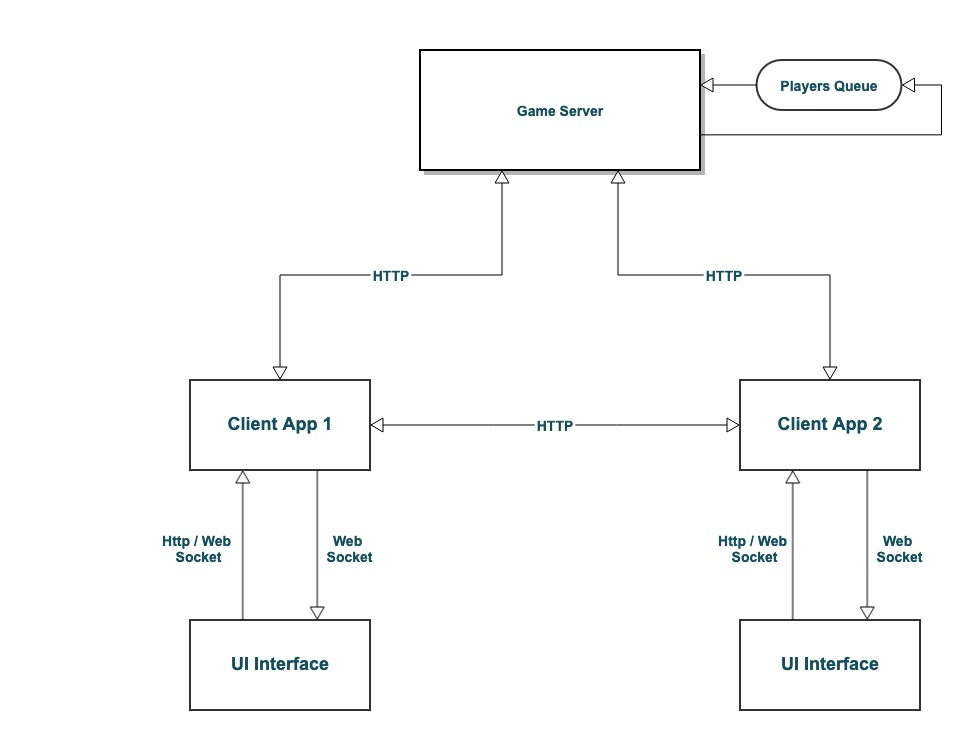
**Assumptions:**

1. There will be a game server where it holds all the player and games information.
2. At one time, two players can play the game. If second player is not available then first player need to wait till second player arrives.
3. User can start a game.
4. User cannot pause/stop the game .
5. As of now there is no way to see how many games a user played and how many he has won and lose. Only through database queries he can find these details.

**Architecture:**



Server Client – Communication



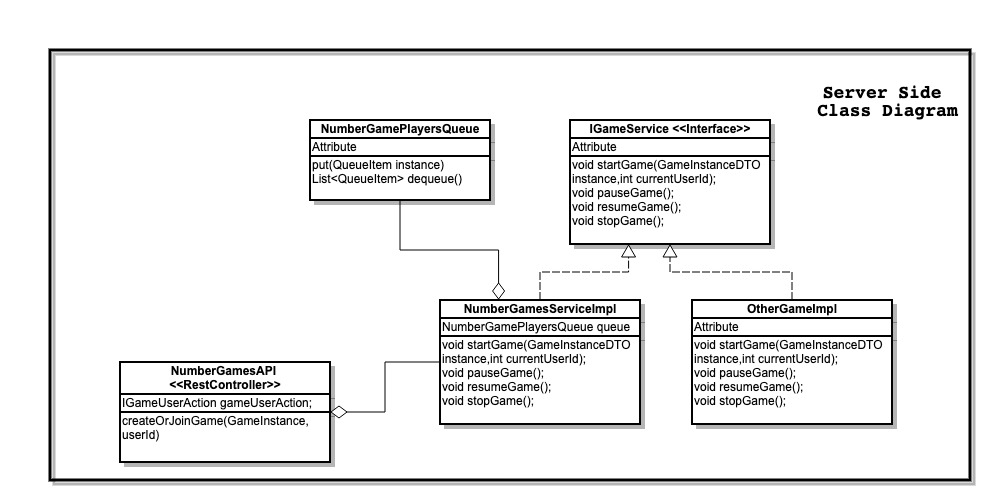
**Game Server:** this is core server which interacts with the database and manages queue for each game which will help in two players to play the game online.

**Client App:** this application normally will ask game server to provide any required data. Also two client app communicate directly via HTTP REST end point. Also it uses web socket to push updated data to UI.

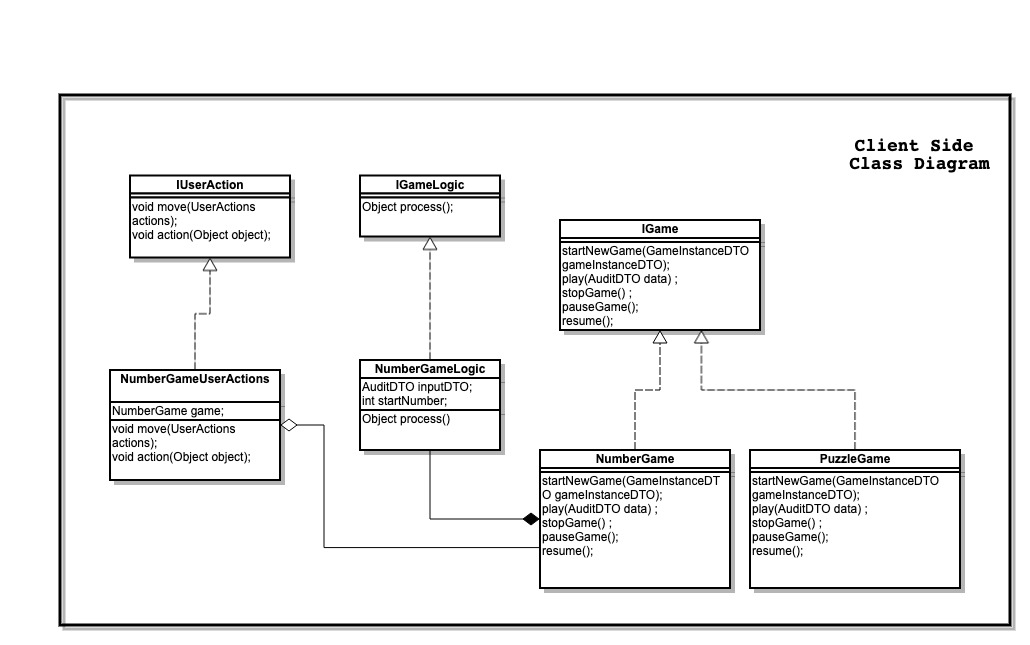
**UI Interface:** this is an angular application which uses http and websocket connection to send and receive data.

**Class Diagram:**

**Server Side class Diagram**

****

**Client side class diagram**

****

**Server Side Interface:**

**IGameUserAction:** this interface having basic methos to start, stop, pause and resume the game. Using these methods user can update the status of the game in db. Currently for Number Game application, user cannot pause and resume the game. User only can start or stop the game.

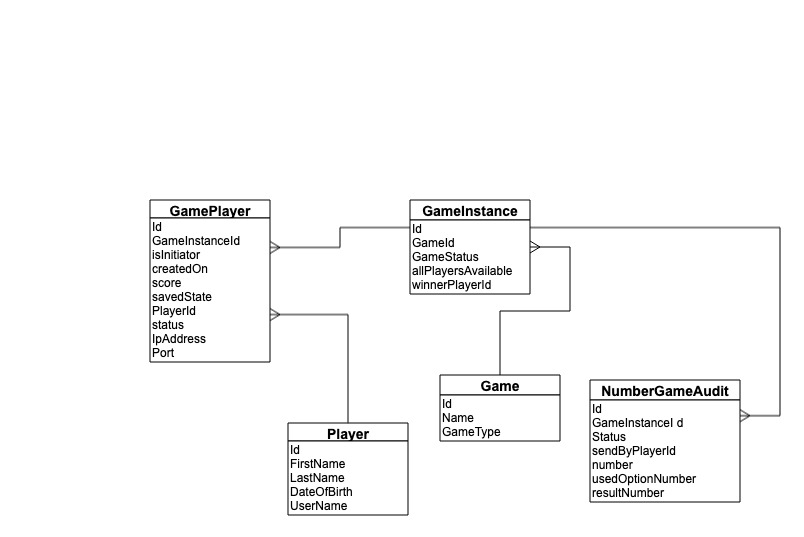
**Client (App) Side Interface:**

**IGame:** This interface is mainly focusing on user action which comes from the UI. Normally it contains methods to start, stop, pause and resume the game

**IGameLogic:** This interface is used to define a logic to process the data. For number game, we implementing this class to add/subtract with 1,0 or -1 to get the number which is divisible by 3 and hence return the result.

**IUserAction:** This interface is used to capture the user event which is fired from UI. For this game there wont be any moves, its contains on the action of pressing any 1 of option out of 3. i.e.. 1, 0 or -1.

Entity diagram:



**Controllers in Server:**

Player: Controller to perform create, update, fetch and delete player details.

Game: Controller to perform create, update, fetch and delete games details. While creating game you need put corresponding list of Play Area (Map).

Game Player: Controller to group a list of player who will be the part of given game instance.

Game instance: Controller to create a game instance, where each instance will be having information regarding game which is being played and its status.

Number Game Audit: this controller will help storing each step of the game.

**Controllers in Client (App) side:**

Manage Game: Controller to fetch game details.

Manage Number Game : Controller to start and stop a game.

Number Rest End Point: Controller to give end points to start/play a game.

**Technologies:**

Server Side: Java, Spring Boot, JPA, Web Socket, Rest API

Database: MySQL

UI: Angular

How application runs:

1. Player 1 request to play a game, his request will go to the server and placed in queue.
2. When player 2 request to play same game, his request also will go to queue, then application will pick these two player and create a game instance object.
3. Once it created game instance, it also puts both player details in Game player table for future reference.
4. Then out of these two player, one player will be assigning a task to initiate the game and a request will go from server to one of client to initiate the first interaction with other client app.
5. From then onward, these two client app will communicate with the user interaction of choosing value like 1, 0 or -1.
6. Finally game will come to an end when any one of client app reaches the number to 1 and that user will be the winner.
7. In All the activity, audit entry will be send to core server.
8. To push the update from game app to UI, I am using web socket.

**Limitations:**

1. Implemented only to start the game, there is no way to pause, resume and cancel the game. Except user can refresh the page to initiate again new game.
2. Make sure before running the Client app, server should be in running state.
3. Data validation not done while saving data to data base.
4. Exception handling not done properly.
5. Logging are missing in the code.
6. Junit testing done only on positive scenario, negative scenario not covered. Only 60% covered on server code, client code is much dependent on server code hence didn’t covered junit for client code.
7. Junit for API not covered.

**Known Issues:**

1. Sometime web-socket connection may not initialize, because of that you wont see data in UI.
2. Right now there is no time limit if only one person starts the game and wait for second player to join.
3. Didn’t much tested the application, you may encounter any bug.
4. No uniqueness validation – example: game name should be unique in system.
5. If a player is already playing a game, then another game session should not create.