Virtual Roulette

Your task is to write a backend service for a virtual roulette game. This Backend application should provide following services:

1. **Sign In service:** User must be able to sign in with username and password. After successful sign in authorized client application should be loaded.
2. **User Balance:** Method to request user’s balance from the client application. Balance must be provided in US dollar cents.
3. **Make Bet:** Method to make bets from client application. User’s bet (selected numbers and bet amount) will be submitted from the client app as a JSON string. After receiving bet from client, bet string must be analysed whether it is correct or not, which can be done by provided *“RouletteBetAnalyzer”* library for java, for C# and PHP please check the appendix.

**Example (**given code is only for java,for C# and PHP please check the appendix**):**

Import the library: ge.singular.roulette.\*

2nd step:

//bet string received from the client

String bet = "[{\"T\": \"v\", \"I\": 20, \"C\": 1, \"K\": 1}]";

//check bet string’s validity

IsBetValidResponse ibvr = CheckBets.*IsValid*(bet);

//as a response you will receive whether bet string is correct or not, and bet amount made by user (in cents).

System.*out*.println("Is bet valid: " + ibvr.getIsValid() + " bet amount in cents is: " + ibvr.getBetAmount());

After validation user’s bet string and bet amount must be stored in the database. Next, by using secure random you should generate winning number (from 0 to 36). Then you should check whether user has won something or not, which again could be done by provided library:

**int** winnum = 12; //winning number, generated by secure random

**int** estWin = CheckBets.*EstimateWin*(bet, winnum); //by calling *EstimateWin method from* CheckBets class you can determine if user won something.

System.*out*.println("User won: " + estWin + " cents.");

// if estWin is equal to 0 this means that user lost.

And finally save winning number and user’s won amount in database, and return result to the user. Result should be returned as Status (whether bet was correct and accepted), Spin ID (automatically generated on the server side), Winning Number and Won Amount in cents.

**Note:** for each bet, server side application should record user’s IP address and date & time when the bet was made.

1. **Game History:** Method to request game history for the user. Game history list returned to the client must contain the following information: Spin ID, Bet Amount, Won Amount and Date & Time (when the bet was made).
2. **Sign-out method:** Server side application must be able to track whether user is connected to the backend and if he/she is active. If user is not active for more then 5 minutes, he/she must be automatically signed out from the system.
3. **Jackpot:** Method for requesting current jackpot amount.

Jackpot is calculated in the following manner: when user makes a bet, 1% from each of his/her bet will be added to the jackpot amount. When jackpot amount is changed it must be automatically pushed to all connected clients.

**Notes:**

* Communication protocol between client and server must be http/https protocol. (Also suggestions (with explanation) about other protocols, that can be compatible with client application written in Javascript, are more then welcome).
* As a database MySQL must be used.
* Configuration information for database must be stored in web.xml file.
* Data transfer footprint between server and client must be as small as possible
* Database connection must be able to handle thousands of simultaneous connections, requests from concurrent users.
* Code must have comments

Appendix

## Pre-generated bet strings for testing purpose:

[{"T": "v", "I": 10, "C": 1, "K": 1},{"T": "n", "I": 11, "C": 1, "K": 1},{"T": "n", "I": 8, "C": 1, "K": 1},{"T": "n", "I": 5, "C": 6, "K": 1},{"T": "n", "I": 19, "C": 1, "K": 1},{"T": "n", "I": 16, "C": 1, "K": 1},{"T": "n", "I": 14, "C": 1, "K": 1},{"T": "s", "I": 18, "C": 1, "K": 1}]

[{"T": "v", "I": 17, "C": 1, "K": 1},{"T": "n", "I": 20, "C": 1, "K": 1},{"T": "n", "I": 23, "C": 1, "K": 1},{"T": "n", "I": 26, "C": 1, "K": 1}]

[{"T": "n", "I": 17, "C": 1, "K": 1},{"T": "n", "I": 19, "C": 1, "K": 1},{"T": "n", "I": 22, "C": 1, "K": 1}]

[{"T": "n", "I": 14, "C": 1, "K": 10},{"T": "n", "I": 17, "C": 1, "K": 10}]

[{"T": "v", "I": 20, "C": 1, "K": 1}]

## Source Codes:

## Java example - Roulette.java

String bet = "[{\"T\": \"v\", \"I\": 20, \"C\": 1, \"K\": 1}]";

IsBetValidResponse ibvr = CheckBets.IsValid(bet);

System.out.println("Is bet valid: " + ibvr.getIsValid() + " bet amount in cents is: " + ibvr.getBetAmount());

int winnum = 12;

int estWin = CheckBets.EstimateWin(bet, winnum);

System.out.println("User won: " + estWin + " cents.");

**Note:** include RouletteBetAnalyzer.jar and json\_simple-1.1.jar library files in a project.

## C# - Roulette.cs

string bet = "[{\"T\": \"v\", \"I\": 20, \"C\": 1, \"K\": 1}]";

IsBetValidResponse ibvr = CheckBets.IsValid(bet);

Console.WriteLine("Is bet valvid: " + ibvr.getIsValid() + " bet amount in cents is: " + ibvr.getBetAmount());

int winnum = 12;

int estWin = CheckBets.EstimateWin(bet, winnum);

Console.WriteLine("User won: " + estWin + " cents.");

**Note:** include ge.singular.roulette.dll dll files in a project.

## PHP - roulette.php

//Include Required Classes

require\_once("classes/IsBetValidResponse.php");

require\_once("classes/CheckBets.php");

$bet = "[{\"T\":\"n\",\"I\":11,\"C\":1,\"K\":1}]";

$checkInstance = new CheckBets();

$validateBet = $checkInstance->IsValid($bet);

$isValid = $validateBet->getIsValid();

$betAmount = $validateBet->getBetAmount();

echo "Is bet valid: ".$isValid." bet amount in cents is: ".$betAmount;

$winnum = 12;

$EstimateWin = $checkInstance->EstimateWin($bet, $winnum);

echo "User won: " . $EstimateWin . " cents.";