Apps GD Costa Rica – Challenge

We are looking for talented individuals to work on research and development projects to create exciting solutions with cutting-edge technologies. As part of that process, we request you to complete the following challenge. This challenge consists on creating a solution to a small problem and then making it publicly available in the web.

Instructions

Challenge duration: 5 days.

This is a brief instructions list, you can find a more detailed explanation in the following sections:

1. Provide a solution for the Rock-Paper-Scissors problem.
2. Create a RESTFull API to interact with the data.
3. Create a web application to provide information about the solution.
4. Upload the code into GitHub or a similar code sharing site.
5. Send the URL of the web site to this email:

   gabriel.zuniga-elizondo@hpe.com

You can use any technology to solve this challenge and any public server to publish it.

Please don't share this document, we trust you!

Problem definition: Rock-Paper-Scissors

In a game of rock-paper-scissors, each player chooses to play Rock (R), Paper (P), or Scissors (S). The rules are: Rock beats Scissors, Scissors beats Paper, but Paper beats Rock. A rock-paper-scissors game is encoded as a list, where the elements are 2-element lists that encode a player's name and a player's strategy.

```
[[ "Armando", "P" ], [ "Dave", "S" ]]
# => returns the list ["Dave", "S"] wins since S>P
```

a) Write a method algorithm that takes a two-element list and behaves as follows:

- If the number of players is not equal to 2, raise an exception.
- If either player's strategy is something other than "R", "P" or "S" (case-insensitive), raise an exception.
- Otherwise, return the name and strategy of the winning player. If both players use the same strategy, the first player is the winner.

b) A rock, paper, scissors tournament is encoded as a bracketed array of games - that is, each element can be considered its own tournament.

```
[
  [
    [ ["Armando", "P"], ["Dave", "S"] ],
    [ ["Richard", "R"], ["Michael", "S"] ],
```

```
    ],
    [
      [ ["Allen", "S"], ["Omer", "P"] ],
      [ ["John", "R"], ["Robert", "P"] ]
    ]
  ]
```

Under this scenario, Dave would beat Armando (S>P), Richard would beat Michael (R>S), and then Dave and Richard would play (Richard wins since R>S); similarly, Allen would beat Omer, Robert would beat John, and Allen and Robert would play (Allen wins since S>P); and finally Richard would beat Allen since R>S, that is, continue until there is only a single winner.

- Write a method that takes a tournament encoded as a bracketed array and returns the winner (for the above example, it should return [“Richard”, “R”]).
- Tournaments can be nested arbitrarily deep, i.e., it may require multiple rounds to get to a single winner. You can assume that the initial array is well formed (that is, there are 2^n players, and each one participates in exactly one match per round).

The first and second place of the championship must be stored into a database, the first place earns 3 points while the second place earns 1 point. The users will always have a unique name, so the same user name could win the first place in two different championships earning a total of 6 points.

REST API

The API must contain mandatory 2 resources and 1 optional. Find the specification of the web services below:

POST api/championship/result

Stores the first and second place of a tournament, each user is stored with their respective scores. The user names will be unique, but the same user can win 1 or more championships. Returns the operation status if successfully.

Resource information:

Resource URL: /api/championship/result

HTTP Method: POST

Response format: JSON

Parameters:

`first` required	The name of the winner of the championship. Example: `Dave`
`second` required	The name of the second place of the championship.

Example: `Armando`

Example Request

POST	`http://<mypublicserver.com>/api/championship/result`
POST Data	`first=Dave&second=Armando`

Example Output

```
{
  status: 'success'
}
```

GET api/championship/top

Retrieves the top players of all championships. Returns the list of player names based on the count provided.

Resource information:

Resource URL: /api/championship/top

HTTP Method: POST

Response format: JSON

Parameters:

`count` optional	Sets the count of records to be retrieved. If not provided, the default value is 10. Example: `123`

Example Request

GET	`http://<mypublicserver.com>/api/championship/top?count=10`

Example Output

```
{
  players: ["Dave", "Armando", "Robert"]
}
```

(Optional) POST api/championship/new

Receives the championship data and computes it to identify the winner. The first and second place are stored into a database with their respective scores. Returns the winner of the championship.

Resource information:

Resource URL: /api/championship/new

HTTP Method: POST

Response format: JSON

Parameters:

`data` mandatory	The data of the championship following the bracketed array standard. Example: `[["Armando", "P"],["Dave", "S"]]`

Example Request

POST	`http://<mypublicserver.com>/api/championship/new`
POST Data	`data=[["Armando", "P"],["Dave", "S"]]`

Example Output

```
{
  winner: ["Dave", "S"]
}
```

Notes:

- The resource URL is a proposal, if for some technology restriction you cannot comply with that specific URL, you can use a different URL.
- The host name used in the URL (mypublicserver.com) must be changed by your own server host name. It could be a plain IP address, it does not have to be a registered domain.

Code sharing

All the code created for this challenge must be open source and publicly available. You can use any web-hosting service such as GitHub or similar. The web-hosting service must allow the source code to be downloaded. The source code published must include all the code created for the algorithm, the web services, the web application and the database scripts.

As stated before, you can use any framework, development language, database engine and server of your preference. However, following coding standards and creating a maintainable, readable and clean code will be rewarded.

Web site

The solution must be published along with a small web application that **must** contain the following information:

- Form to upload a text file (.txt) containing the definition of a championship. A file can only contain 1 championship.
- Reset button to clean up the database and start over.
- Provide your own example of files containing championships. These must be downloadable files. The championships must follow the bracketed array standard as described in the problem definition.
- A list with the top 10 players from all championships registered.
- The REST API documentation (the same included in this document), but providing the real resource URL. This documentation must be clear enough to allow any developer test the API.
- The URL to where the source code is publicly available (GitHub or similar).
- A short description about yourself, a description about the solution implemented and a justification of the technologies used.

There is no specific design that the web application must follow, however a good user experience in the web application and any extra useful information will be rewarded.

Please send the URL of the web site to the email stated in the instructions section.