Mukto Software - SureCash Recruitment Screening (Question)

Kitty is a space voyager from Zephyrus Galaxy. Last month her spaceship landed in Earth. She is curious about the new world. But she can’t ask people randomly because it will draw NASA’s attention which will lead to a terrible experience for her. She knows that most human are not much amicable to Aliens. Kitty somehow found a generous friend in Mukto Software Ltd and asked for his help. You know how much busy we are at Mukto! Now it’s your duty to build a bot that can satisfy Kitty’s quest.

Technically speaking, you should expose a REST API that will take a question via an **URL Encoded GET parameter** in the mentioned url endpoint and will response the answer in JSON format. Base url of your api should be the heroku app url you’ve provided during registration.

The bot should be able to answer following questions:

**Greetings**

Kitty is a polite alien. She can exchange some greetings. Don’t be rude on her.

Sample:

“Hello! How are you?”

“Hi! What is your name?”

“Good morning! I am Kitty! It’s a pleasure to meet you!”

Her greeting messages will contain “Hi!”,”Hello!”,”Good morning/evening/night!” and you can tell her anything according to your interest that starts with the term “Hello, Kitty!”

Route: GET /greetings

Params: q=question (url encoded)

Example: /greetings?q=Hello!%20How%20are%20you?

Response: JSON

{

“answer”:”Hello, Kitty! rest\_of\_your\_answer”

}

**Weather Info**

Kitty is curious about our planet’s weather. She may ask for general weather related information for a city like - Current temperature, humidity, Weather Forecast of today.

Sample question:

“What is today’s temperature in <City Name (i.e. Dhaka/London)>?”

“What is today's humidity in <City Name>?”

“Is there Rain/Clouds/Clear weather today in <City Name>?”

Sample answer:

“24 C” or “297.15 K”

“Rain”

“Yes” / “No”

Route: GET /weather

Params: q=question (url encoded) i.e. ?q=anything from above

Response: JSON

{

“answer” : ”your answer”

}

**Basic World Affairs**

Queries on famous people, country, movies, books, music etc. You can get help from Wikipedia or google to answer these questions.

Sample question:

“Who is the president of Bangladesh?”

“How old is Putin?” (in years)

“How long is Titanic movie?” (in minute)

“What is the prime language of Bangladesh?”

“Who is the current CEO of Google?”

It is normal that you can’t build an Oracle over the night that knows everything. Just try to answer Kitty’s question. If you can’t answer something, tell her - ***“Your majesty! Jon Snow knows nothing! So do I!”***Save this question to your memory. Show Kitty the list of questions that you have failed to answer when she says - ***“Tell me! What don’t you know?”***

The format of the answer should be **<#>*<Serial No><space><Question><New Line>***

Route: GET /qa

Params: q=question (url encoded)

Example: /qa?q=who+is+bill+gates%3F

Response: JSON

{

“answer” : ”your answer”

}

**Comments on Solution**

* For weather info there are a lot of api’s. [Open Weather Map](http://openweathermap.org/api) is one of them.
* There are a lot of ways to solve the third problem. Tough one is to parse Wikipedia/Google query result. There are some python packages to parse wiki. Without help of any framework or helping library this is quite tough to complete within 24 hours. There are some quick solutions too. [Quepy](http://www.machinalis.com/blog/quepy/) is a framework to convert natural language question to [SPARQL](http://www.w3.org/TR/rdf-sparql-query/) query. Using SPARQL we can extract information from [DBpedia](http://wiki.dbpedia.org/). One ugly but working implementation is given below.

var sparqlQuery="";

$.ajax({

url: "http://quepy.machinalis.com/engine/get\_query",

data: {

question: "Who is Tom Cruise?"

},

type: "GET",

dataType: "json",

success: function(data){

sparqlQuery= data.queries[0].query;

console.log(sparqlQuery)

},

error: function(data){

console.log(data)

}

});

$.ajax({

url: "http://dbpedia.org/sparql",  
 data: {  
 "debug": "on",  
 "timeout": "0",  
 "query": sparqlQuery,  
 "default-graph-uri": "",  
 "format": "application/sparql-results+json"  
 },  
 type: "GET",  
 dataType: "json",  
 success: function(data){  
 console.log(data);

//code for extracting output from the json  
 },  
 error: function(data){  
 console.log(data)  
 }

});

* There are other services / 3rd party api’s that can help you to solve problem 3. Let’s see whether you can find any of them :)