**Motivation**

Everything we do at Uber is about location. More specifically, it’s about moving people from one location to the next in the most convenient, reliable, and efficient way possible. In order to accomplish this goal, we must quickly dispatch our clients to the nearest available cab - and herein lies your challenge.

**Overview**

Should you choose to accept it, your challenge is to build an HTTP server that can quickly find the nearest cabs to a particular client location. Utilize python and you can use a web service framework of your choice. Your server must be able to handle the following CRUD operations:

1. *Create / Update request*

Insert a new record or update the latitude and longitude of a particular cab

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | URL | Returns | Normal Response |
| PUT | /cabs/(cab\_id) | No body | 200 OK |

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Required | Name | Type | Default | Description | Example |
| Yes | latitude | Float |  | GPS coordinate | 37.799476 |
| Yes | longitude | Float |  | GPS coordinate | -122.511635 |

1. *Get request*

Get the full details of the cab

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | URL | Returns | Normal Response |
| GET | /cabs/(cab\_id) | Cab | 200 OK |

**Parameters**

This action does not have any parameters.

1. *Query request*

Search for nearest cabs. The returned data does not have to be sorted and should be a list of cab records. DIstance should be calculated based on the [Haversine for mula](http://en.wikipedia.org/wiki/Haversine_formula).

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | URL | Returns | Normal Response |
| GET | /cabs | Cab | 200 OK |

**Query Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Required | Name | Type | Default | Description | Example |
| Yes | latitude | Float |  | Client GPS coordinate | 37.799476 |
| Yes | longitude | Float |  | Client GPS coordinate | -122.511635 |
|  | limit | Integer | 8 | The total number of cabs to limit the response to | 25 |
|  | radius | Float |  | The maximum distance (in meters) from the client location for a cab | 1000 |

**Sample Request**

GET /cabs?latitude=37.763658&longitude=-122.427521=&radius=1000&limit=15

**Sample Response**

[  
 {  
 "id": 23706134,  
 "latitude": 37.788654783559,  
 "longitude": -122.50747748978  
 },  
 {  
 "id": 61344818,  
 "latitude": 37.778952285851,  
 "longitude": -122.43865835511  
 },  
 {  
 "id": 19485186,  
 "latitude": 37.778665475753,  
 "longitude": -122.39094602609  
 },

...  
]

1. *Destroy request*

Destroy a cab

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | URL | Returns | Normal Response |
| DELETE | /cabs/(cab\_id) | No body | 200 OK |

**Parameters**

This action does not have any parameters.

1. *Destroy all request*

Destroy all cab records

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | URL | Returns | Normal Response |
| DELETE | /cabs | No body | 200 OK |

**Parameters**

This action does not have any parameters.

**Evaluation / Additional Notes**

Beyond these basic API specs, the challenge is left intentionally open-ended. The are no requirements for which coding language, database, or design you must use, but we do expect that you can defend the decisions you make. The best solutions will be those that go beyond something that “just-works” and have thought about the implications of real-world production environment. For example, which endpoints do you think will be used most frequently?

All latitude and longitude coordinate are based on the [WGS84 coordinate system](http://en.wikipedia.org/wiki/World_Geodetic_System) (the same one used by GPS enabled devices).

Bonus points if your solution includes a front-end for this API.