DNS Cache

Instruction

DNS or Domain Name System is the phonebook of the internet. DNS translates domain names to IP addresses so browsers can load internet resources. There are a lot of steps involved in this translation process, but in this challenge, we simplified the steps into:

1. Check if domain name is available in cache.
    - If it is available in cache, it’s considered a `cache hit` and DNS returns the IP address to the user.
2. Else, check in the lookup table. This is considered a `cache miss`.
    - If the domain name is available in the lookup table, update the cache and return the IP address to the user.
    - Else, return null. This is considered an `invalid` request.

Your cache should follow LRU (Least Recently Used) algorithm. The LRU cache is a cache eviction algorithm that organises elements in order of use. In LRU, as the name suggests, the element that hasn’t been used for the longest time will be evicted from the cache.

You should provide 2 API endpoints as follows:

1. `<teamUrl>/instantiateDNSLookup`
   When this endpoint is called, your code should store a map of domain names and their IP addresses in a persistent storage i.e files, db. You can use any storage of your choice.

Request:

```
{
  lookupTable: {
    'google.com': '1.2.3.4',
    'amazon.com': '2.3.4.5',
    'yahoo.com': '3.4.5.6',
    'bing.com': '4.5.6.7',
    'facebook.com': '5.6.7.8',
    'instagram.com': '6.7.8.9'
  }
}
```

Response:

```
{
 "success": true
}
```

2. `<teamUrl>/simulateQuery`
   When this endpoint is called, your code should simulate user queries and determine if each query results in a `cache hit`, `cache miss`, or `invalid` request.

Request:

```
{
  cacheSize: 3,
  log: [
    'google.com',
    'google.com',
    'amazon.com',
    'instagram.com',
    'google.com',
    'bing.com',
    'instagram.com',
    'burger.com'
  ]
}
```

Response:

```
status: 200
JSON: [
  { status: 'cache miss', ipAddress: '1.2.3.4' },
  { status: 'cache hit', ipAddress: '1.2.3.4' },
  { status: 'cache miss', ipAddress: '2.3.4.5' },
  { status: 'cache miss', ipAddress: '6.7.8.9' },
  { status: 'cache hit', ipAddress: '1.2.3.4' },
  { status: 'cache miss', ipAddress: '4.5.6.7' },
  { status: 'cache hit', ipAddress: '6.7.8.9' },
  { status: 'invalid', ipAddress: null }
]
```

Explanation:
Based on the example request body provided, cache size is currently set to 3. The table below shows the expected behaviour of the LRU algorithms.

URL Queried	Response	Cache Status Explanation	Cache Content after Query
'google.com'	`{ status: 'cache miss', ipAddress: '1.2.3.4' }`	`cache miss` : 'google.com' is absent in cache but exists in lookup table. Now 'google.com' is added to the cache.	`{ 'google.com': '1.2.3.4' }`
'google.com'	`{ status: 'cache hit', ipAddress: '1.2.3.4' }`	`cache hit` : 'google.com' now exists in cache.	`{ 'google.com': '1.2.3.4' }`
'amazon.com'	`{ status: 'cache miss', ipAddress: '2.3.4.5' }`	`cache miss` : 'amazon.com' is absent in cache but exists in lookup table. Now 'amazon.com' is added to the cache.	`{ 'amazon.com': '2.3.4.5', 'google.com': '1.2.3.4' }`
'instagram.com'	`{ status: 'cache miss', ipAddress: '6.7.8.9' }`	`cache miss` : 'instagram.com' is absent in cache but exists in lookup table. Now 'instagram.com' is added to the cache.	`{ 'instagram.com': '6.7.8.9', 'amazon.com': '2.3.4.5', 'google.com': '1.2.3.4' }`

URL Queried	Response	Cache Status Explanation	Cache Content after Query
'google.com'	`{ status: 'cache hit', ipAddress: '1.2.3.4' }`	`cache hit` : 'google.com' exists in cache.	`{'google.com': '1.2.3.4', 'instagram.com': '6.7.8.9', 'amazon.com': '2.3.4.5' }`
'bing.com'	`{ status: 'cache miss', ipAddress: '4.5.6.7'` }	`cache miss` : 'bing.com' is absent in cache. 'amazon.com' was evicted from the cache because it was the least recently queried domain name.	`{'bing.com': '4.5.6.7', 'google.com': '1.2.3.4', 'instagram.com': '6.7.8.9' }`
'instagram.com'	`{ status: 'cache hit', ipAddress: '6.7.8.9' }`	`cache hit` : 'instagram.com' exists in cache.	`{'instagram.com': '6.7.8.9', 'bing.com': '4.5.6.7', 'google.com': '1.2.3.4' }`
'burger.com'	`{ status: 'invalid', ipAddress: null }`	`invalid` : 'burger.com' is absent in DNS lookup. Cache stays the same.	`{'instagram.com': '6.7.8.9', 'bing.com': '4.5.6.7', 'google.com': '1.2.3.4' }`

Testing

To test your solution in a testing environment, you can call `<challengeUrl>/test` and pass the following request body:

```
{
   "teamUrl": "<insert your team url here>"
}
```

Please take note that your challenge should be deployed in a testing environment instead of localhost to use this endpoint.