

Amazon Light Map

CAB432 - Assignment One

Cian O'Leary n9727442

Introduction

Amazon Light Map aims to help mitigate the ultimate bottleneck on the user experience in the modern world, the speed of light. More so the time it takes light in a fibre connection to reach a server. This is achieved by allowing users to input locations and then calculating which Amazon Region is best due to average distance from the users. Ultimately this tool can provide non-trivial insights into alternative placements in simple situations and can provide a visually representation of placement for more complex situations.

Services

Google Geocoding API

Was used to retrieve the longitude and latitude of the non normalised text input from the user.

Amazon

Amazon has an endpoint to programmatically retrieve their IP ranges for regions. An assumption that was made was that if multiple ranges are used for a region only the last is used and since the order is not consistent this will change over time.

IP Info API

Used to retrieve the longitude and latitude of the IP ranges retrieved from Amazon. Originally I had intended to use the ip vigilante api, however a few days before submission their SSL certificate expired which caused the site to stop functioning so I moved to IP Info.

Use Cases

Use Case 1

As a developer I would like to be shown the closest amazon region to me to reduce upload times while developing. *Appendix 1* shows that a user can simply enter their suburb or address and they are shown the closest region on the map.

Use Case 2

As a business manager I want to be given a short list of regions so I can asses the cost before deciding. *Appendix 1* shows that an ordered list of the top five regions is displayed to the user. This could form the starting point for an investigation into cost and service availability in these regions.

Use Case 3

As a solutions architect I would like to be shown the placement of my users and the approximation of the best region so I can confirm and investigate further. *Appendix 2* shows a user can enter as many or as few locations as they like and it will return a short list and display the top pick on the map

Use Case 4

As an Arctic researcher I would like to know where is the closest region to minimize upload time on my large data sets. *Appendix 3* shows that the program has full support for the world's least populated continent.

Limitations and Technical Discussion

Amazon Light Map is an node express app currently which works as a prototype. Express has two endpoints an index get which returns the static HTML and a post search endpoint. Currently the application can handle around 15 simultaneous searches before the IP Info api throttles.

This could be resolved by caching the list of region lat/lng as new regions are particularly rare. From this point the next bottleneck is the Google API Free tier which is limited to 50 simultaneous requests from the client, however the premium tier has no listed limit.

Our next bottleneck after caching and API limits will be the server instance that the container is running on, however some of this load could be reduced by removing the static index page from the container and hosting it in a service such as AWS S3 with a CDN in front of it to reduce costs. By removing those initial requests and caching the API lookups we should reduce the load on the server significantly allowing for significant scale throughput. In order to scale further we would need to utilise a load balancer or move to a service such as AWS Lambda.

Docker

Node boron is the node 6 LTS (6.11.3), however with node:carbon officially starting LTS next month it may be worth upgrading at that point in time as with Node 8 boasting an average 28% speed increase over Node 6 is more than enough for it to be a worthwhile venture.

Port 8080 could be changed, however you would need to reflect this on line 7 of index.js. Node:Carbon will also have yarn bundled as an alternative default package manager so when that upgrade is made it would make sense to replace `npm install` with simply `yarn`

```
FROM node:boron

WORKDIR /usr/src/app

COPY package.json .
RUN npm install

COPY . .

EXPOSE 8080

CMD [ "npm", "start" ]
```

Testing, Limitations and Compromises

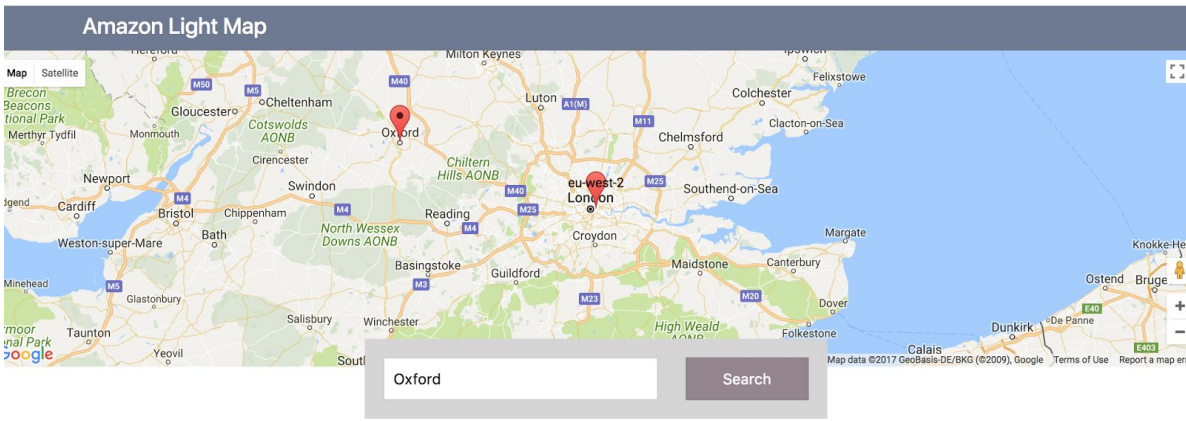
While testing for a final submission a problem was encountered. The problem was the IP Geolocation service that I was using (IP Vigilante) had it's SSL certificate expire which caused the service to stop working. To solve this I moved to a competing service (IP Info).

Possible Extension

One possible extension would be the ability to assign a weight to individual locations. The reason for this would be if 90% of your users are in singapore, 5% in Sydney and 5% in melbourne the system will currently recommended sydney, however in that case it would make much more sense to use the region (ap-southeast-2) which in singapore itself.

Appendix 1

Amazon Light Map



LOCATION INFO:

Longitude:

Latitude:

Name:

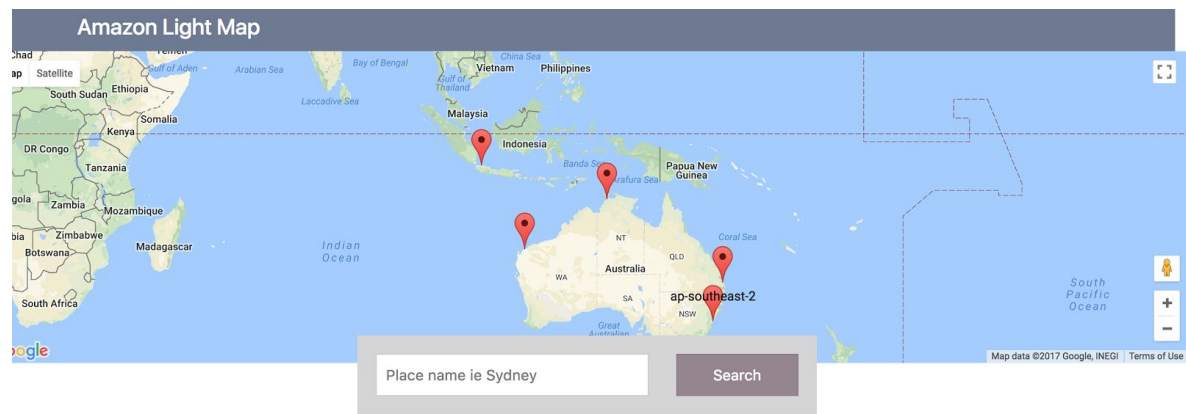
LIST OF PLACES:

- Oxford, UK

CLOSEST ZONES:

1. eu-west-2
2. eu-west-1
3. eu-west-3
4. eu-central-1
5. ca-central-1

Appendix 2



LOCATION INFO:
Longitude: 153.02512350000006
Latitude: -27.4697707
Name: Brisbane QLD, Australia
[Add Location](#)

LIST OF PLACES:

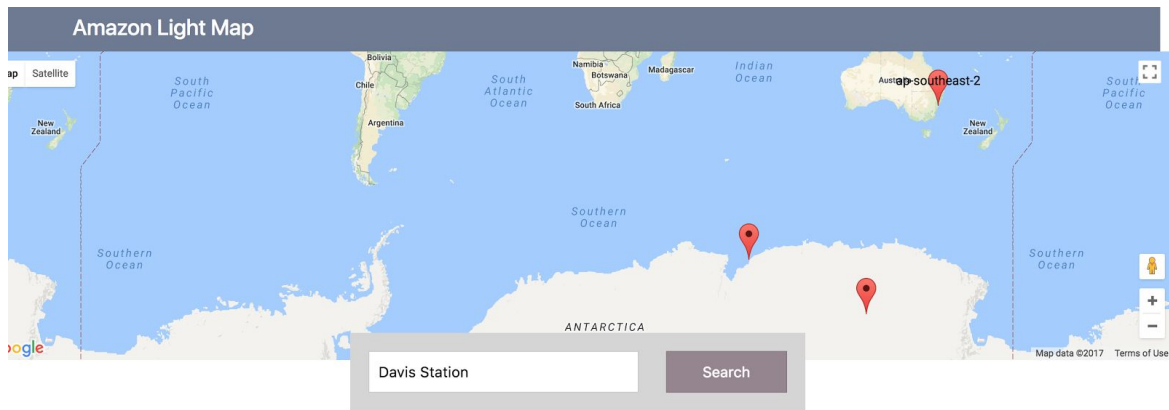
- Darwin NT, Australia
- Onslow WA 6710, Australia
- Jakarta, Indonesia
- Brisbane QLD, Australia

[Search Location](#)

CLOSEST ZONES:

1. ap-southeast-2
2. ap-southeast-1
3. ap-northeast-1
4. ap-northeast-2
5. cn-north-1

Appendix 3



LOCATION INFO:
Longitude:
Latitude:
Name:
[Add Location](#)

LIST OF PLACES:

- Concordia Station, Antarctica
- Davis Station, Antarctica

[Search Location](#)

CLOSEST ZONES:

1. ap-southeast-2
2. ap-southeast-1
3. sa-east-1
4. ap-south-1
5. ap-northeast-1