Decision Point Address Project Report:

Project Requirements and Deliverables:

(Written by Decision Point Senior Engineer and Architect Stanis Billy)

Requirements:

1. Address validation: given a human-readable address string, we should be able to validate whether it is a valid address or not. Extras:

- Being able to specify an address' (and/or its components) format. For example, 5 or 9-digit zip codes, full or 2-digit states, etc.
- In addition to validating, being able to receive a "standardized/correct" address in a particular format.

2. (Forward) geocoding: given a human-readable address string, we would like to get its geographic coordinates (latitude/longitude). Extras:

a. Reverse geocoding. Given geographical coordinates, get its human-readable address representation. There are some reservations on this front, as entering the same coordinates for a particular address into a service may not give that same exact address.

Expected Deliverables:

The outcome of this project should be a single (or multiple) executable script and an accompanying documentation. The script should accept both single and batch inputs, with outputs formatted accordingly. Batch inputs should produce batch outputs, and single (stream) inputs should produce single outputs. The two parts of the project should have distinct endpoints/scripts. Notes on these deliverables below:

• Working code and technical demo
  - Code can be hosted in a public repository

• Documentation
  ▪ Detailed breakdown of various 3rd party services explored
  ▪ Sources/URLs
  ▪ Pros/cons
  ▪ Free option? Features/limitations?
  ▪ Paid option? Features/limitations?
  ▪ Hosted option?
  ▪ Batch request?
  ▪ Single request?
  ▪ BAA option (for HIPAA)?

o Any 3rd party module used should be listed and documented.

Project Outcomes, Analysis and Completed Deliverables:

(by developer Will Murphy)

Working Code: https://github.com/Will-Murphy/dp-address-project

This code satisfies both main requirements 1 and 2, in both stream and batch format, with extras of address standardization and reverse geocoding.

Description: A command line tool for processing stream/batch address data for validation, standardization and two-way geocoding. Provides an internal interface for connecting too and implementing third party services to process address data and produce desired output data.

High Level Details: This project was designed for allowing interchangeable use of third party services so that it will work and be easy to use with different third party services if they are to be changed in the future. Also, if the desired outputted standardized address format were to be changed, it could be done without affecting how the overall program works. See the next steps in the README for more details on the implementation and future of this project.

Demo, Technical Documentation, Selected Third Party Module Details:

See README for Most of These Details:
https://github.com/Will-Murphy/dp-addre2ss-project/blob/master/README.md

By following the setup and usage instructions, a technical demo working for all requirements is available after cloning the repo. Also, third party modules used are listed, documented and their details noted - more on that below.

Third Party Services Used and Research on Available Options:

See Address Service Comparison Worksheet for full comparison of third party details:

Third Party Forward Geocoding and Address Validation Service: SmartyStreets
(https://smartystreets.com)

Relative Benefits and Reasoning for Initially choosing:
SS is fast, relatively cheap (with the unlimited subscription) and easy to use. Furthermore, it provides in depth analysis and does geocoding and validation in one request. It meets all our requirements except reverse geocoding. In contrast, some companies provide all services for all requirements (Loquate, Service

Objects) but in all different services/ requests or make you pay a premium for single service use (assumed only by the naming of those services as premium). Also, those companies don't show publicly available pricing, and there are not many downsides to using a mix of services. Additionally, reverse geocoding was an extra for this project, so it doesn't seem like a big disadvantage - especially since there are other specialized services who can do it better and cheaper if needed. This, combined with its ease of use, ease of testing and also credibility, due to its use by companies like Netflix and Zillow, made it stand out during preliminary research, and was why I chose to use it in development. There were many other validation services available online, besides those in the worksheet and mentioned, but those provided less info, credibility and/or pricing details to use for relative comparison as well as often lacking free trials for me test with.

Concerns:
After implementing and testing though, some problems started occurring with the Smarty API not recognizing valid addresses. Overall, for smarties relatively low price, and the functionality it provides, it seems worth trying to use it and solve these initial problems. No other services provide validation and geocoding in one request for a low cost, and if it is relied on by companies like Netflix and Zillow, it seems reasonable that these initial issues can be overcome, potentially with some of the solutions mentioned in the README next steps section. If this problem can be solved, it is my opinion that Smarty is still the best option. Otherwise, using Loquate or Service Objects, which provide everything needed (though not in one package, and for an undisclosed amount) may be good potential choices (a rep from Service Objects called me and was telling me their validation is more accurate than smarty streets).

Third Party Reverse Geocoding Service: OpenCageData
(https://smartystreets.com)

Reasoning for Initially choosing: Smarty doesn't provide reverse geocoding, so a second service was needed - which seems okay because it doesn't seem like we will need as much reverse geocoding in comparison to forward geocoding and validation. Initially, I went with OpenCage more because it was easy to use, and had a good, accurate free option, rather than because of its relative benefits. I mainly wanted to get something down to show multiple services could be used together modularly, and it happened to add reverse geocoding. Overall, It is a decent option as it is reasonably priced and seems to be fairly accurate, but there are issues.

Concerns: Moving forward Geocodio (see worksheet) seems like it would be a better option than OpenCage, as it is not based on OpenStreetMap data, which though has proven to be good in limited testing, is open source. Also, there is

only a monthly subscription available, which may get expensive on top of another service like Smarty Streets. Additionally, OpenCage, has a very restrictive rate-limited response whereas none of the paid tiers of Geocodio are rate limited , and it allows batch jobs. Moreover ,Geocodio allows you to pay by volume, is cheaper than Google's volume pricing, and has an optional unlimited subscription for the same price as OpenCage. Similar to OpenCage it has an easy to use API and accompanying python sdk that would make implementing it as an additional service very easy. Finally, it is relied on by companies Amazon, Comcast and Aetna so it is fair to assume it has a good baseline of accuracy and reliability.