MBTA Routes/Stops Design Document

Problem Statement

This project aims to interface with the MBTA APIs to collect and display certain statistics about MBTA routes and stops. In particular, we will only be dealing with the Light Rail and Heavy Rail Routes as they are called, which can also be thought of and referred to as the “subway”. The information we will want to collect will be related to the: Names of the subway routes, names and relationships of the routes to stop that they make, and how to leverage the interactions between the routes and stops to produce plans to get from a certain stop to another, through these routes.

Technical Resources

API Documentation

The documentation for the MBTA APIs can be found here:
https://api-v3.mbta.com/docs/swagger/index.html
With the following information extracted as it pertains to our use-case.

Routes

Routes can be found at /routes and have the following fields (filtered for only our use case):

```
{
  "data": [
    {
      "id": "string",
      "attributes": {
        "long_name": "Red Line"
      }
    }
  ]
}
```

And accept the following query param to filter by type:
filter[type]
With the following mappings:

Value	Name	Example
0	Light Rail	Green Line

1 Heavy Rail Red Line

Stops

Stops can be found at /stops and have the following fields (filtered for only our use ase):

```
{
  "data": [
    {
      "id": "string",
      "attributes": {
        "name": "Parker St @ Hagen Rd"
      }
    }
  ]
}
```

And accepts the following query param to filter by route:
filter[route]
We will need to call this stops endpoint per-route as far as I can tell, as there is a include option which includes route, but it has the following disclaimer (which means we cannot call with multiple routes and include the route information in the response body):

Note that route can only be included if filter[route] is present and has exactly one /data/{index}/relationships/route/data/id.

Assumptions Made

Green Line

I am assuming that the different lines within the Green Line should be treated differently, as they are treated differently by the API, and they do reach different stops.

Taking a Route to the Same Stop

My route-finding implementation currently does not disallow taking a route to the same stop you are already one in some cases. This could be improved, but given the time constraint I have not addressed it yet.

Outline of Implementation

Question 1

We will call the endpoint: https://api-v3.mbta.com/routes?filter[type]=0,1 and collect the name off of each route that has been pre-filtered by the MBTA web-server. I feel that having the web-server filter with the type-filtering will be easy-enough to implement, while remaining clear, and will save us bandwidth.

Question 2

We will get the stop information for each route, by calling https://api-v3.mbta.com/stops?filter[route]=ROUTE.ID and collect it:

- Keyed by the route to see the route with the fewest and most number of stops
- Keyed by the stop to see which stops have more than one route

Question 3

Will will collect the stop and route information, again keyed in each direction:

- Keyed by stop to each route to know which routes we can reach from a given stop
- Keyed by route, to know which stops to try to keep exploring for each route

The algorithmic approach will be a recursive implementation that starts at the start stop and explores each route connected to that stop, for each route, it will then start to explore each stop on that route, keeping track of where it has already been to not explore the same routes multiple times. In pseudo-code the naive solution would be:

```
func explore(stop-to-routes, route-to-stops, start, end, routes):
    if start == end:
        return routes, true
    for route in stop-to-route:
        for subStop in route-to-stop:
            if !routes.contains(route):
                path, isSolution = explore(
                    stop-to-routes,
                    route-to-stops,
                    subStop,
                    end,
                    routes.with(route))
                if isSolution:
```

path, isSolution

return nil, false