Uwave Challenge

Aim

To plan and build a bus-timing service that provides API endpoints for frontend engineers to build upon by formatting data from an external API provider.

Use Cases

1. As a user of the service, I need the timings of specific bus services at specific bus stops.
2. I would also like to see the available buses that pass through my stop.
3. I would like to see the available routes that pass through my stop.

Features

1. Input validation where if the bus stop does not exist, the program will not proceed.
2. GetEndpoint.go file reads the JSON from the specified bus stop URL and extracts the stop's information.
3. Basic API to handle HTTP requests.

Assumptions

Assumptions:

- I assumed that the information provided by the external API provider is accurate in terms of bus timings at the respective bus stops.
- I assumed that having the bus timing in minutes would be more intuitive for the user and hence I converted the timings provided by the API to minutes.

Sample Output

```
Bus Stop Number:
383050
Bus 31410 on Campus Loop Red (CL-R) is arriving in  4  minutes.
```

Originally I wanted to add more information in the API endpoint:

```
type BusTimings struct {
    StopID   string `json:"stop_id"`
    StopName string `json:"stop_name"`
    BusLines []struct {
        BusLineID   int    `json:"busline_id"`
        BusLineName string `json:"busline_name"`
        Buses       []struct {
            VehicleID       int    `json:"vehicle_id"`
            ArrivalTime     int    `json:"arrival_time"`
        } `json:"buses"`
    } `json:"buslines"`
}
```

but due to time constraints I decided to simplify the output to just a few important variables without nesting anything.

Bonus Part

a. To overcome this rate limit on calling the external API, I would store API calls for bus timings of the same bus and would update it every 1 minute. This solves the issue as lesser API calls are needed when there are multiple requests for the same bus timing. As my program returns bus timing in minutes, updating the timing of the bus every minute is sufficient for it to be accurate to the nearest minute. Due to time constraints I was unable to implement this.
b. To manage inaccurate bus timings we can calculate the estimated time for the bus to cover the distance between the user's bus stop and the current location of the bus by using location and speed information of the bus provided by the external API provider.