Problem Statement

We are going to build an application that is meant to be a sort of status checker for some common websites that exist online. The application continuously polls the status of these websites and exposes APIs to retrieve the information.

Requirements (Must have)

1. **The application should expose an API using which we can submit lists of websites to monitor**
    - Implement an HTTP server and expose it on a port eg: 3000
    - Expose an API endpoint (hint: POST /websites) and accept the list of websites in the request body as an array
    - Save the list of websites in an in-memory map.
2. **The application should monitor the status of all submitted websites every 1 minute**
    - Create a go routine which loops over all the websites and checks if they are responding to HTTP traffic (hint: status code 200). If yes, the website can be considered as UP, otherwise DOWN. Once the status check is done for all websites, sleep for 1 minute and continue this process forever.
    - Status checks of N websites can be done concurrently using goroutines to improve performance.
    - The status of each website can be saved in the same in-memory map where the list was stored.
3. **The application should expose an API using which we can see the status of all websites. It should also support passing the name of a specific website and then it should only return the status of that particular website**
    - Expose an API endpoint (hint: GET /websites) which returns an array of websites with their current status
    - Support a query parameter in API (hint: GET /websites?name=www.facebook.com) which then returns the status of the given website

Requirements (Good to have)

1. Unit test cases for the 2 APIs (APIs mentioned in Requirements #1, and #3)
2. Use of Golang interface to check the status of the website. Today we are relying on the HTTP status code of a website to determine whether it's up or not. Tomorrow, we could use some external third-party service for the same. Hence we could write up an interface to do the status check. Currently, it will use the HTTP implementation which can be switched later to advanced implementation. Below given is a sample for your reference. However, feel free to use your own names and signature.
3. Use of Go packages is encouraged.

```go
type StatusChecker interface {
    Check(ctx context.Context, name string) (status bool, err error)
}

type httpChecker struct {
}

func (h httpChecker) Check(ctx context.Context, name string) (status bool,
err error) {
    // your implementation to check status using HTTP call
    return
}
```

Sample Input (for POST /websites**)**

```
{
        "websites" : ["www.google.com","www.facebook.com",”www.fakewebsite1.com”]
}
```

Sample Response (for POST /websites**)**

200 OK

Sample Input (for GET /websites**)**

Query param (optional) ?name=www.facebook.com

Sample Response (for GET /websites**)**

```
{
        "www.facebook.com" :  "UP",
        "www.google.com" :      "UP",
        "www.fakewebsite1.com": "DOWN",
}
```

Demo

1. You need to submit a short video (Preferably less than 5 minutes) to demonstrate your work.
2. There is no UI/Frontend work required for this Project, hence please use the CURL or Postman tool to call the APIs and show the output.
3. Please add logs in the application to show that it is performing status checks on the website and then you can also print results to demonstrate the behavior of the Status Checker go routine.