# Programming Assignment 2

## Goal

- Get familiar with HTTP protocol. Understand how web API works.
- More practice on socket API.

## Description

In this assignment, you will design and implement a dynamic web server that provides the following services.

### /api/evalexpression

This API is to help clients evaluate arithmetic expressions and return them the results. The client is going to send an arithmetic expression in an HTTP `POST` request. See the example below where client wants to evaluate expression `7+9-11+6`.

```
POST /api/evalexpression HTTP/1.0\r\n
Content-Length: 8\r\n
\r\n
7+9-11+6
```

And your server should return the following HTTP response.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/html\r\n
Content-Length: 2\r\n
\r\n
11
```

Note: this API is should only support simple arithmetic expressions: (a) only involve integers, (b) only support '+' and '-' operators. For all unsupported arithmetic expressions, your server should return an HTTP 400 response (bad request).

## /api/gettime

This API is to help clients get the local time on the server. The client is going to send an HTTP `GET` request. See the example below.

```
GET /api/gettime HTTP/1.0\r\n
\r\n
```

And your server should return its local time in a human readable string format. For example, you can return a response like below.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/html\r\n
Content-Length: 24\r\n
\r\n
Sun Sep 29 07:41:37 2019
```

You have the freedom to choose the time format you like.

## /status

This page should the status information of your web server. It should contain

- The number of API calls for (`evalexpression` and `gettime`) during the last minute, last hour, last 24 hours, and lifetime.
- The most recent 10 expressions clients submitted to evaluate.

Your server need to return a valid HTML page that is able to render successfully inside a browser. For instance, the page you return can look like

```
......
<h1>API count information</h1>
<h3>/api/evalexpression</h3>
<ul>
  <li>last minute: 2</li>
  <li>last hour: 10</li>
  <li>last 24 hours: 128</li>
  <li>lifetime: 314</li>
</ul>
<h3>/api/gettime</h3>
<ul>
  <li>last minute: 1</li>
  <li>last hour: 2</li>
  <li>last 24 hours: 2</li>
  <li>lifetime: 7</li>
</ul>
<h1>Last 10 expressions</h1>
<ul>
  <li>7+8-11</li>
  <li>1+1+1+1+1+1+1</li>
  ......
</ul>
......
```

You don’t need to worry about persisting historical count, just need to collect this stats since the start of the server is sufficient.

### Others

For all other URLs, your server need to return HTTP 404 response. (e.g. /api/whatisthis, /index.html, etc.)

## Requirements

- You can only use the socket API (in module `socket`). You are not allowed to use any other higher level modules like `requests`, `urllib`, etc.
- Your server should be able to handle both HTTP/1.0 and HTTP/1.1 requests.

## Grading policy

100 points in total.

- [20 pts] Correctly parsing HTTP request and sending HTTP response
    - E.g. read/write socket according to the HTTP protocol, extract Content-Length, etc.
- [30 pts] Implement /api/evalexpression
    - Correctly parse the expression from HTTP POST request.
    - Able to evaluate expressions only involving integers, ‘+’, and ‘-’ correctly.
    - Correctly send HTTP response.
    - For unsupported expression, correctly send HTTP 400 response.
- [10 pts] Implement /api/gettime
- [30 pts] Implement /status
    - Can correctly show accumulated stats for /api/evalexpression and /api/gettime
    - Can correctly show the latest 10 expressions clients sent.
- [10 pts] For other URLs, return HTTP 404 response

## Submission instruction

Please your code and report to all TAs and cc z.sun@northeastern.edu by the deadline.