Outlab 6 : Git & WebStack

Please refer to the general instructions and submission guidelines at the end of this document before submitting.

P1. Up and running [65 points]

One of the more popular system architectures is the client server model, where in a machine acts as a server and other(s) as client(s). The responsibility of server is to reply to requests generated by client(s). A request generally consists of a data to locate the server and some questions that a client has. The data regarding location of the server is used by Networking elements (like routers, switches etc..) to guide the request to the server. Once a request reaches the server, the questions asked by client are processed and answered back in a response. Response (similar to request) contains location of the client and the answers to the questions.

In server client architecture, one of the more popular & classic request-response model is the webpage model. In webpage model a request is generally a string that contains both the location of server and the questions (generally in terms of parameters). And a response is a web page. This is where HTML (Hypertext markup language) comes into picture. You see HTML is used to create web pages. So a webpage response is actually HTML script. HTML is a markup language. HTML is not a programming language. There is no concept of variables/data containers/if else/loops in HTML.

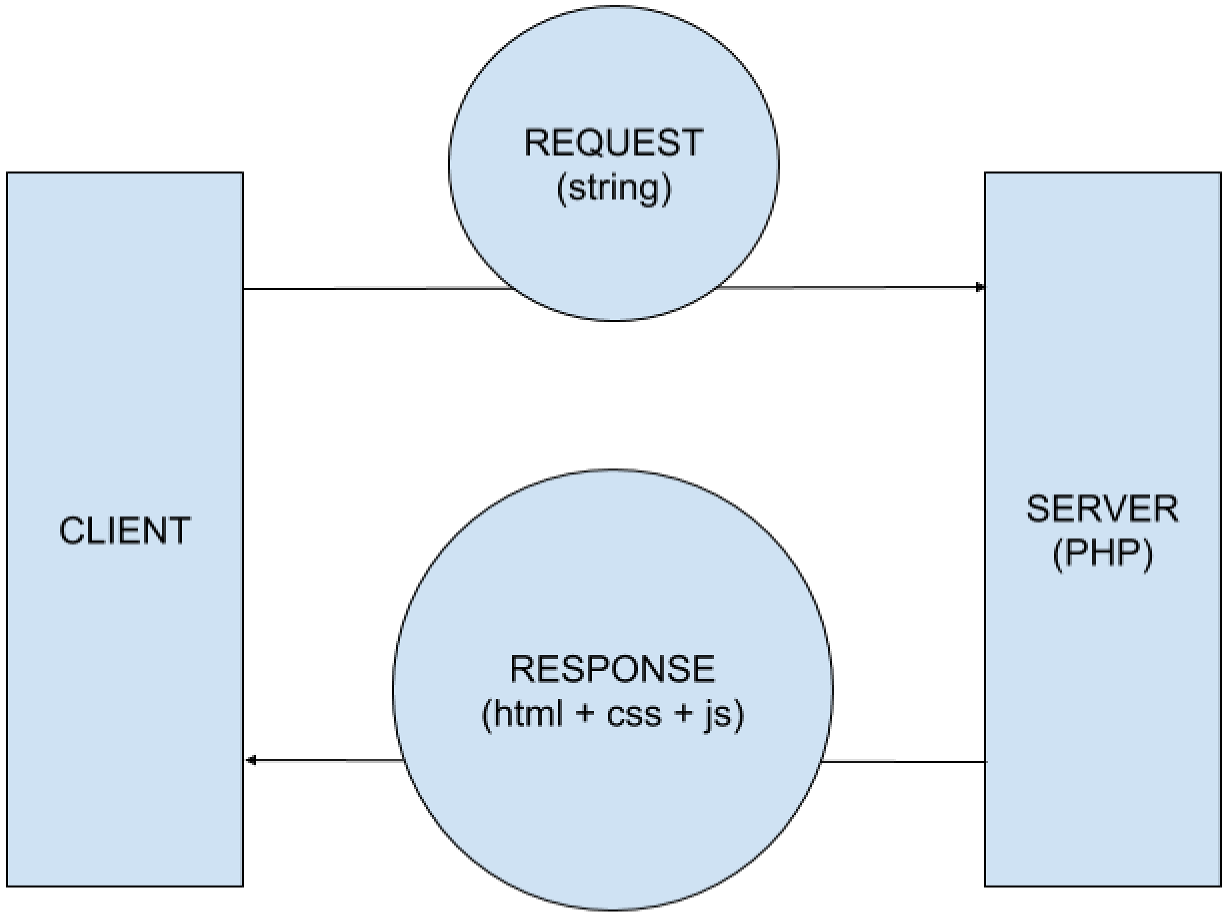
HTML (for that matter most of WebStack) is highly autodidactic. This is one of the better references for the entire webstack which has inbuilt tutorials.

HTML itself is sufficient for representing data. But over time a need for sophistically organizing and visualizing data was felt and one of the products of that is CSS (Cascading Style sheets). As the name suggests a CSS script styles a page generated by HTML script. This is an example of a webpage where almost no CSS is used. This is an example of a webpage where CSS is used properly. This is an example of web page where CSS is not used properly.

HTML and CSS by themselves are enough to generate and organize data. But a webpage created solely by HTML and CSS is static (just a bunch of strings). To bring life into a webpage JS (Javascript was introduced, not to be confused with Java). JS is a full on turing complete programming language. It can do stuff. Mainly JS is used for manipulating HTML and CSS content in the context of webstack. Thus it brings a sense of dynamism to a webpage.

Note that all of the above discussed scripts HTML CSS and JS are all part of response. When the response reaches client, client (generally a browser) interprets it and displays what you daily see as web pages.

The following diagram summarizes what all is discussed until now



PHP (Hypertext Preprocessor) (also a full on turing complete programming language) is tailor made for creating responses to client requests. Typically a server side (backend developer) writes a PHP script which accepts parameters (questions) from client and outputs an appropriate response contains HTML CSS & JS scripts.

Task 1

In this task you are going to build your homepage. This is going to stay with you for the rest of your IITB life. If people want to know something about you the first thing they do is to checkout your homepage (um... maybe facebook first, but your homepage shall have extensive details).

1. Your homepage should be a reflection of you. So it should typically includes a short brief about yourself, your interests, your projects, your ambitions and goals, your public contact information
2. To start off first create a webpage that is accessible using the url (`www.cse.iitb.ac.in/~<cseuserid>/`, referred to as **root url** from here on).
3. This page should contain
    a. Your name.
    b. **A one line description about yourself**, who are you as a person?
    c. When clicked on your one line description a **modal** should popup, which shows your biography.
    d. A **carousel** of your public pictures.
    e. A page hit counter which shows number of times the page was accessed. This is a measure of popularity of your page. It should (obviously) increase by 1 for every access. This can be done by storing a counter in a separate file, reading from file, sending the counter in web page, incrementing it by 1 and storing it back.
4. Make a page showing **your interests, ambitions and goals, your hobbies, your favorites** etc…
    a. You can use **dropdown/tabs** to neatly represent each section
5. Make one page for your **projects** (done, ongoing, future planned) etc…
6. Make one page for your **CS251 team**
    a. team members, and their profile pictures
    b. team name and why you choose that name
    c. a logo of your team
    d. a real-time clock showing current data and time refreshing every second
7. Make a page which shows your **contact information**. This will be public so be careful.
8. Make a page about your academics.
    a. Use CSS **accordions** and **collections** to group the courses you took under the semesters you did them.
    b. Create your current timetable using **table** tag
9. Make a page which contains a **form** containing fields **Name** and **Comment and** a **submit button**.
    a. Anyone who comes to this page can add a comment about your homepage and click submit.

    b. Submission should not occur if any of the fields are empty.
    c. All such comments must be saved in a file.

10. Make a page which reads all the comments in the file in the previous task and displays them in a neat **collection**.
11. A page/webpage in this task may refer to a html or php file based on context and use.
12. Give titles to each page appropriately. Give a common icon (which is displayed to the left of title. Icon should not be inside body. For example google.com has a little G icon) for all pages.
13. All pages should contain (the same) **menu bar** which contains links to all web pages you are going to create. You can give any names for your pages/files.While grading we will first go to your root url and use menu bar to navigate between pages. So make sure you connect all web pages in menu bar and keep your website up.
14. Style everything neatly. Bootstrap and Materialize CSS are good frameworks for that.
15. As you can see this task is largely subjective. The marks will be given on aspects including but not limited to
    a. Content
    b. Functionality
    c. Aesthetics
    d. Interesting features

Task 2

1. Write **true** or **false** (not True, TRUE, true. etc...) for each of the following one question per line in that order in **trueorfalse.txt**
    a. PHP is dynamically typed language.
    b. Javascript is statically typed language.
    c. PHP has duck typing
    d. JS does not have duck typing.
    e. PHP is a compiled language.
    f. JS is a compiled language.
    g. PHP does not have types.
    h. JS does not have types.
    i. HTML is a compiled language
    j. A php file can contain segments of C++ code in it.

Task 3

Go to your facebook account and open one of your post. Use inspection tool in your browser to change the number of likes to 100K. Take a screenshot of the modified post and name it **fb.png.** Does this mean the #likes are changed permanently?

General Instructions

- Make sure you know what you write, you might be asked to explain your code at a later point in time
- Grading may be done automatically, so please make sure you stick to naming conventions
- The deadline for this lab is **Sunday, 26th August, 23:55.**

Submission Instructions

After creating your directory, package it into a tarball **`<rollno1>-<rollno2>-<rollno3>-outlab6.tar.gz`** in ascending order. Submit once only per team from the moodle account of smallest roll number.
The directory structure should be as follows (nothing more nothing less)

```
<rollno1>-<rollno2>-<rollno3>
    └── P1
        ├── fb.png
        └── trueorfalse.txt
```