**Assignment 2: Implement a basic http server with api services.**

**Total points: 30**

**Part 1: Basic HTTP functions (10 Points)**

In this assignment, you will implement a basic http server which responds to the clients’ HTTP requests, i.e., GET, POST, PUT, and DELETE HTTP requests. For each request, the server performs an appropriate action and returns a response to the client.

In part 1 of the assignment, you will implement a basic HTTP server which will respond to clients’ GET requests. A GET request asks the server for some resource (e.g., an HTML file) usually specified as a URL, and the server returns the specified resource (e.g., contents of the HTML file requested) as an HTTP response.

In this client-server communication process, the client opens a TCP connection with the server. It then sends an HTTP request to the server. In response, server returns the contents to the client. The client receives the HTTP response from the server, displays the response to the standard output. The connection is closed by the server after sending the HTTP response. This means a single connection is used only for a single HTTP request-response operation.

***HTTP request and response format.*** An HTTP request contains a header and optionally a body. An HTTP response also contains a header and optionally a body. The header and body contain one or more lines of texts. There is a blank line between the header and body. The following shows a simple HTTP request containing only the header and no body:

GET / HTTP/1.1

The above request has one line in the header and there is no body. The fist line of header denotes that it is a GET request and asking for a server resource indicated by “/” and HTTP protocol to be used “HTTP/1.1.” The server may respond to this request by sending the contents of a default HTML file (index.html) to the client as below.

HTTP/1.1 200 OK

Content-length: 26

<html>

Hello world

</html>

The above HTTP response contains two lines in header and three lines in the body. The header contains HTTP protocol used in the response and a status code (with status code meaning) that denotes the response status. The body is the contents of the *index.html* file. The second header line specifies the length (in bytes) of the body (included in the HTTP response).

As you can see above, a HTTP response header contains a status code (e.g, 200, 403, 404) which signifies a special meaning describing the status of the response. Below, we show the basic status codes sent by a HTTP server.

|  |  |
| --- | --- |
| **Status code** | **Meaning** |
| 200 OK | Request served successfully. For example, requested resource was found in the server |
| 403 NOT FOUND | Requested resource was not found in the server. |
| 404 BAD REQUEST | Request is not in proper HTTP request format. |
| 201 CONTENT CREATED | Some contents have been created in the server database with respect to a POST request of a client. |
| 204 NO CONTENT | Some operatios (update/delete) was done successfully. This is a status code to be sent with respect to a PUT/DELETE request. |

***Basic HTTP functions.*** You will implement the server which should respond to the following client-requests.

|  |  |  |
| --- | --- | --- |
| **Request type** | **Response** | **Comments** |
| GET / HTTP/1.1 | Sends the *index.html* file to the client.  Response format:  HTTP/1.1 200 OK  Content-length: XX  <contents of index.html> | XX indicates the length (in bytes) of the response body. |
| GET /file\_path HTTP/1.1 | Sends the specific html file to the client whose location is specified by a path location.  Response format:  HTTP/1.1 200 OK  Content-length: XX  <contents of index.html> | Example.  Http request:  GET /contact.html HTTP/1.1  Http response:  HTTP/1.1 200 OK  Content-length: XX  <contents\_of\_file> |
| GET /file\_path HTTP/1.1 | Response format:  HTTP/1.1 404 NOT FOUND  Content-length: 0 | This response is sent when the request resource (file) is not found on the server. |
| GET | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 | This response is sent when the request is malformed. For example, in this request the request URL is not given. |
| GET / | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 | This is a malformed request as the HTTP protocol is not specified in the request header. |
| Any other malformed request | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 | Please see the above row. |

For implementation details, please run the executable files provided to you.

**Part 2: API functions (20 Points)**

The server contains a database (in a file named api.db) of user information. This file stores records of users (an id, a name, and an email address of the user). Please see the database file (api.db) for details. The server retrieves records, creates new records, updates and deletes existing records in this database with respect to clients’ GET, POST, PUT, and DELETE HTTP requests. In all api requests, the path field in the HTTP request header must start with “/api” which indicates that it’s an api request (not a basic GET request).

The functions to be implemented as part of this assignment are discussed below. Please run the executable files for more information.

***Basic API functions.*** The server should perform the following API functions.

|  |  |  |
| --- | --- | --- |
| **Request type** | **Response** | **Comments** |
| GET /api/2 HTTP/1.1  Or  GET /api HTTP/1.1  id=2 | Retrieves the record from the server api database whose id is 2 to the client. The response format is given below.  Response format:  HTTP/1.1 200 OK  Content-length: XX  id=2&name=tanvir&email=tanvir@gmail.com | Record is to be fetched from the database (a file). |
| GET /api/12 HTTP/1.1  Or  GET /api HTTP/1.1  id=2 | Response format:  HTTP/1.1 404 NOT FOUND  Content-length: 0 | This response is sent when the request resource (record whose id = 12) is not found on the server database. |
| POST /api HTTP/1.1  name=sukarna&email=sukarna@gmail.com | Creates a record in the server api database for the given name and email. Sends the following response to the client.  Response format:  HTTP/1.1 201 CONTENT CREATED  Content-length: 0 |  |
| POST /api HTTP/1.1  name=sukarna | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 | This response is sent when the request is malformed. For example, in this request the email of the user is not given. This response must be sent to the server for other similar malformed requests. |
| PUT /api HTTP/1.1  id=2&name=alim&email=alim@gmail.com | Updates an existing record in the api database whose id=2. The name and email fields are updated for the record whose id=2.  Response format:  HTTP/1.1 204 NO CONTENT  Content-length: 0 | This is a successful update request. |
| PUT /api HTTP/1.1  id=22&name=alim&email=alim@gmail.com | Response format:  HTTP/1.1 404 NOT FOUND  Content-length: 0 | This is an unsuccessful update request because the user id is not found in the api database. |
| DELETE /api HTTP/1.1  id=2 | Deletes the record from the database whose id=2. Sends the following response to the client.  Response format:  HTTP/1.1 204 NO CONTENT  Content-length: 0 | This is a successful delete operation. |
| DELETE /api HTTP/1.1  id=12 | Response format:  HTTP/1.1 404 NOT FOUND  Content-length: 0 | This is an unsuccessful update request because the user id is not found in the api database. |
| PUT /index HTTP/1.1  id=22&name=alim&email=alim@gmail.com | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 | This is a malformed api request. Api requests must be sent to URL /api. |
| Any other malformed request like the above | Response format:  HTTP/1.1 400 BAD REQUEST  Content-length: 0 |  |

***Client behavior.*** Your HTTP client receives a HTTP request from the user (stdin), creates a connection with the server, and sends the request to the server. The response returned from the server is received and displayed to the user (stdout). The client loops for another request from the user. In this assignment, the HTTP server closes the connection after serving each request. Thus, clients must create another connection for the next request. Usually, a user will type multiple lines of texts as HTTP request. If the “>” symbol is entered by the user in a new line, then this marks the end of the user input (http request). Please run the executable files for more details.

Code base. You are given with three files:

* *network\_helper.cpp*: This file implements socket related operations which you are already familiar with from your first assignment.
* *http\_helper.cpp*: This file implements some basic HTTP related functions such as extracting header and body from a HTTP request. You will use these functions in your program. This file also contains an important function for text tokenization. For details, please see the http\_helper.cpp.
* *api\_helper.cpp*: This file implements all the required api related functions. The functions retrieve records from the database, creates, updates, and deletes records in the database. Thus, you are not required to perform file read/write operations yourself. All the operations are already implemented in the *api\_helper.cpp* file. All you need is to call appropriate functions of the api\_helper.cpp from your program.