Introduction:  
This exercise entails the implementation of RFC 2616 in C++11 utilizing sockets. The web-server executes asynchronous tasks using C++ selectors.

Testing:  
Postman was employed for conducting tests, with each request accompanied by a corresponding test within the collection provided alongside this submission.

Code:  
The primary objective was to eliminate strings. Consequently, upon receiving each message, it is promptly parsed into a self-implemented HttpRequest object.  
Subsequently, the request is processed, generating an *HttpResponse* object. Prior to transmission via the socket, the *HttpResponse* object undergoes parsing into a valid HTTP response string.

The object-oriented perspective of this implementation involves several key classes designed to facilitate its functionality:

1. **Uri**: This class defines a URI, featuring a single string property named "path."
2. **SingletonHtmlPlaceholder**: Implementing the Singleton design pattern, this class ensures the creation of a single instance of a string.
3. **HttpMessage**:

• Enumerations for *HttpVersion*, *HttpMethod*, *HttpStatusCode*, *QLanguage* (representing the "lang" query parameter), and utility functions for converting strings to enums/class instances and vice versa.

• **QueryParams**: A container for a map<string, string> of query parameters.

• **HttpMessageInterface**: Comprising class members such as content, headers and version, this interface automatically updates the Content-Length header when content is set, initially set to 0.

• **HttpRequest**: Implementing the *HttpMessageInterface*, this class also includes class members such as method, params, and URI.

• **HttpResponse**: Also implementing the *HttpMessageInterface*, this class adds the status code class member.

1. **HttpServer**:

* Defined types for a generic HTTP request handler: *HttpRequestHandler\_t = std::function<HttpResponse(const HttpRequest&)>*.
* A collection is established to store all server request handlers: *map<Uri, map<HttpMethod, HttpRequestHandler\_t>>*, facilitating the *HandleHttpRequest* function's determination of when to return *NotFound* or *MethodNotAllowed*.

1. **SocketService**: This class encompasses all functions relating to the sockets utilized by the *HttpServer*.