ABSTRACT

The Hypertext Transfer Protocol (HTTP) is a crucial part of the internet communication system, serving as the foundation for transmitting data over the World Wide Web. The aim of this project is to thoroughly examine the design, architecture, and implementation of HTTP. The study will start by giving an overview of the protocol's evolution, starting with HTTP/1.0 and ending with the current version, HTTP/2, while highlighting the advancements made in each iteration. The different components of HTTP will be analyzed in detail, including its methods (GET, POST, etc.), status codes, headers, and cookies, to provide a clear understanding of how these elements work together to allow for effective communication between clients and servers.

Additionally, the project will delve into the security concerns surrounding HTTP, such as cross-site scripting, cross-site request forgery, and man-in-the-middle attacks, and the measures that can be taken to counteract these threats, such as encryption. The project will conclude by summarizing the current state of HTTP and exploring future developments and trends, including new technologies like HTTP/3, which aim to improve the protocol's performance and security. The end result of this project will be a comprehensive understanding of HTTP and its significance in the modern web.

METHODOLOGY

To demonstrate the benefits of HTTP 1.1 over HTTP 1.0, you can create a simple experiment in Node.js.

1.Create two Node.js servers, one for HTTP 1.0 and another for HTTP 1.1.

2.Use Benchmark tester tools, or a similar tool, to send multiple requests to each server and measure the response time.

3.Compare the response time for both servers and observe the difference.

by this we can prove that HTTP 1.1 server sets the Connection header to keep-alive, which allows multiple requests to be sent over a single connection.using same method we can prove how http/2 is better than http/1.1It is important to note that HTTP/2 provides multiplexing, allowing multiple requests to be sent over a single connection, and it uses a binary format for more efficient data transferusing some request methods we can show the importance of multiplexing in http/2