Department of Computer Science and Engineering,
Anna University, Chennai- 600025
CS23502- NETWORKS AND DATA COMMUNICATION
Practical
July- Oct 2026
Year/Sem/Batch : III/V/ N

Exercise: 3	Applications using HTTP protocol	23- 07- 2026

Observation questions:

1. What is HTTP, and is it a connection-oriented or connectionless protocol? Which transport-layer protocol does it run over?

2. What is the difference between HttpURLConnection and the newer java.net.http.HttpClient introduced in Java 11?

3. What are the common HTTP request methods (GET, POST, PUT, DELETE, HEAD)? Explain the difference between GET and POST.

4. What is the structure of an HTTP request message (request line, headers, blank line, body)? What is the structure of an HTTP response message (status line, headers, blank line, body)?

5. What do HTTP status codes like 200, 301, 404, and 500 signify? Which category (2xx, 3xx, 4xx, 5xx) does each belong to?

Execution Questions (Programming Language - Java)

1. Write and run a Java program using HttpURLConnection (or HttpClient) to send a GET request to a public URL (e.g., a test API) and print the response status code and body. Modify the program to print all the response headers returned by the server.

2. Write a Java program to send a POST request with a JSON body to a test endpoint and display the server's response.

3. Set up a simple HTTP server in Java (using com.sun.net.httpserver.HttpServer) that responds with "Hello, World!" for GET requests on /hello. Run it and access it from a browser or another Java client program.

4. Use the Java program to fetch a URL that does not exist (e.g., a wrong path) and observe/record the status code and message returned (should be 404). Modify the client program to measure

and print the time taken to complete the HTTP request-response cycle.

Spot Questions

1. Modify your Java HTTP client program to send the GET request with a custom request header (e.g., User-Agent: MyLabClient/1.0). Run it against a test URL and demonstrate, using the response or a header-echoing test service, that the custom header was actually sent to the server.

2. Modify your Java HTTP server program to respond differently based on the request method — return one message for a GET request on /hello and a different message (or a "405 Method Not Allowed" style response) if a POST request is made to the same path. Demonstrate both cases working correctly.