جامعة بيرزيت
BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

ENCS3320-Computer Networks

Project#1 due 14/12/2025

1- **This is a group project, so you are allowed to work in groups of max 3 students**
2- **Use socket programming to implement the communication between the browser and the webserver (do not use frameworks like flask or Django).**
3- **You can use libraries like hashlib**
4- **Important: Each screenshot should include the date and time of your computer.**

This project consists of three main parts. Part1 focuses on some computer networking tools. Part 2 requires implementing client and server applications using TCP and UPD. Part3 demonstrates how a web server can be created using Python's low-level socket programming, without relying on web development frameworks. The server communicates with web browsers using the HTTP protocol, serves static web pages (HTML/CSS/PNG/JPG), and supports secure user registration and login. The system also implements session management using cookies, which allows the server to remember authenticated users across browser tabs within the same session.

You have to submit

1- **A report in pdf format (only pdf format) on moodle (itc.birzeit.edu) that contains Screenshots** with **detailed explanation, codes, runs, etc.**
2- The code with comments (include the code in the pdf file and as text file .py or .java or .c as well)
3- You are allowed to send compressed file (e.g., .zip). But you must send a complete report as pdf file separately.

Part1:

1- In your own words, what are ping, tracert, nslookup, and telnet (write one sentence for each one)
2- Make sure that your computer is connected to the internet and then run the following commands:
   1- Ping a device in the same network, e.g. from a laptop to a smartphone
   2- ping 1.1.1.1
   3- tracert 1.1.1.1
   4- nslookup 1.1.1.1
3- use wireshark to capture some DNS messages.

Provide a screenshot of the runs and brief explanation of the output.
From the ping results, what can you say about the distance to 1.1.1.1

Part2:

Using socket programming, implement TCP and UDP client and server applications in go, python, java or C. The server should listen on port 8090. **The client sends the word "START" using TCP and then sends numbers from 0 to 1000,000 using UDP (each number is in a different packet) and then finally sends "END" using TCP. The Server should count the numbers received and how often the order of the packets was wrong and display this information on the screen.**

Part3:

Using socket programming, implement a simple but a complete web server in go, python, java or C that is listening on port 8099.

Have a look also on rfc2616 (https://datatracker.ietf.org/doc/html/rfc2616)

0- from rfce2616, explain 5 terms related to Cache

The user types in the browser something like http://localhost:8099/ar or http://localhost:8099/en

The program should check

1- if the request is **/ or /index.html or /main_en.html or /en (for example localhost:8099/ or localhost:8099/en)** then the server should send main_en.html file with Content-Type: text/html.

The main_en.html file should contain

HTML webpage that contains

a. "ENCS3320-My Tiny Webserver 25/26" in the title
b. "Welcome to our course **Computer Networks, This is a tiny webserver" (**part of the phrase is in **Blue)**
c. Use CSS to make the page looks nice
d. Divide the page in different boxes and put student's information in the different boxes
e. Include CSS as a separate file
f. Summarize point 0 above in a box
g. Group members names and IDs
h. Some information about the group members. For instance, projects you have done during different course (programming, electrical, math, etc), skills, hobbies, etc.
i. The page should contain at least An image with extention.jpg and an image with extension .png
j. A link to a register.html file
k. A link to login.html

    l. a link to https://www.w3schools.com/python/python_strings.asp

2- If the request is /ar then the server response with main_ar.html which is an Arabic version of main_en.html
3- if the request is an .**html** file then the server should send the requested html file with Content-Type: text/html. You can use any html file.
4- if the request is a **.css** file then the server should send the requested css file with Content-Type: text/css. You can use any CSS file
5- if the request is a .**png** then the server should send the png image with Content-Type: image/png. You can use any image.
6- if the request is a **.jpg** then the server should send the jpg image with Content-Type: image/jpeg. You can use any image.
7- Use the status code 307 Temporary Redirect to redirect the following
    a. If the request is /chat then redirect to chatgpt website
    b. If the request is /cf then redirect to https://www.cloudflare.com/ website
    c. If the request is /rt then redirect to ritaj website

8- If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html)
    1- "HTTP/1.1 404 Not Found" in the response status
    2- "Error 404" in the title
    3- "The file is not found" in the body in **red**
    4- Your names and IDs in **Bold**
    5- The IP and port number of the client
9- Register .html it enables users to register a new account by providing username and password. The password is hashed and stored in file data.txt (use sha256)
10- Login.html: During login, the server hashes the entered password and checks it against the stored hash. If valid, the server sends protected.html (the browser displays this page). Then the server generates a unique session ID randomly, stores it in memory, and sends it to the browser inside a cookie. When the user navigates to other pages or opens new browser tabs, the browser automatically sends the session cookie back. The server checks the stored session table and grants access to protected pages like protected.html. when the user clicks on logout the session is removed on the server side, making further access require login again.
11- The program should print the HTTP requests on the terminal window (command line window).

Provide screenshots of the browser with brief descriptions to show that your project works as expected. Test the project from a browser on the **same computer** and from **a different computer or phone**.

Provide also a screenshot of the HTTP request printed on the command line.