Vietnam National University – Ho Chi Minh city
University of Science
Faculty of Computer Science

COURSE PROJECT

COMPUTER NETWORKING

PROJECT of SEMESTER I
2020 – 2021

SOCKET PROGRAMMING

Class: 19CLC10
Instructor: Chung Thùy Linh

Full name	Students ID	Work	Accomplished
Trương Gia Đạt	19127017	_Socket server programming. _Web programming (HTML, CSS).	100%

Ho Chi Minh , 2020

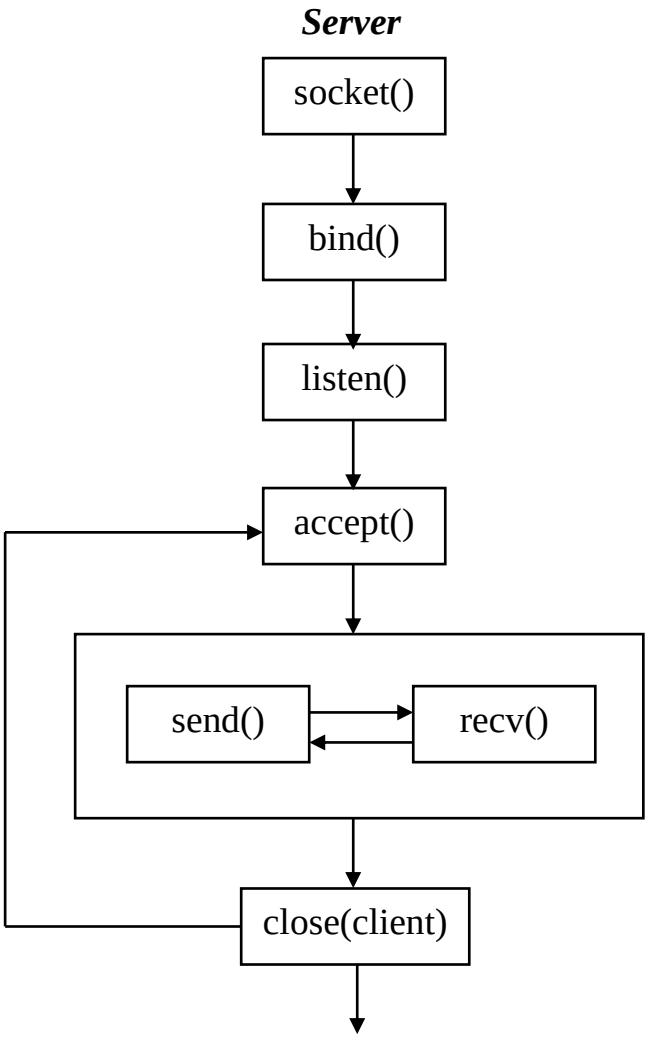
CONTENTS

INTRODUCTION

- In this Lab you will be introduced to socket programming at a very elementary level. Specifically, we will focus on TCP socket connections which are a fundamental part of socket programming since they provide a connection oriented service with both flow and congestion control. What this means to the programmer is that a TCP connection provides a reliable connection over which data can be transferred with little effort required on the programmers part; TCP takes care of the reliability, flow control, congestion control for you. First the basic concepts will be discussed, then we will learn how to implement a simple TCP client and server.

Programming with Sockets - Server

- ***Outline of TCP Server:***



- ***Windows Specific Functions***

  All the methods for declaring and using sockets are available in two header files:

```
#include <WinSock2.h>
#include <WS2tcpip.h>
```

  We will also need to link the `Ws2_32.lib` library in our project settings.

```
#pragma comment(lib, "Ws2_32.lib")
```

  The first thing we have to do before being able to declare or use a socket is make a call to the `WSAStartup()` method.

```
int WSAAPI WSAStartup(
  WORD      wVersionRequested,
  LPWSADATA lpWSAData
);
```

  This method must be called first before any other calls involving the sockets library. This method simply allows the programmer to define the **version** of the Windows sockets specification to be used and stores the result in the `WSADATA` struct.

- ***Creating a Socket***

```
SOCKET WSAAPI socket(
  int af,
  int type,
  int protocol
);
```

  _af stands for address family. For IPv4 we would use the flag **AF_INET** and for IPv6 the flag is **AF_INET6**.

  _type can be defined as either **SOCK_STREAM** for TCP or **SOCK_DGRAM** for UDP.

  _protocol can be set as **NULL**. The caller does not wish to specify a protocol and the service provider will choose the protocol to use.

  _If no error occurs, socket returns a descriptor referencing the new socket. Otherwise, a value of **INVALID_SOCKET** is returned.

- ***Binding a Socket***

```
int WSAAPI bind(
  SOCKET         s,
  const sockaddr *name,
  int            namelen
);
```

_We have declared out socket but if we plan to receive messages using it then we need to bind to it using the `bind()` function. The purpose of binding is to associate a local address with a socket in order to interact with each other.
_Moreover, the server always needs to call the bind function as it is going to listen for new connection requests and serve them.
_If no error occurs, bind returns **zero**. Otherwise, it returns **SOCKET_ERROR.**

- ***Listening for Connections***

_For our server, having already specified our address information, created a socket and bound to it, we now need to listen for new connections requests from clients.

```
int WSAAPI listen(
  SOCKET s,
  int    backlog
);
```

_After all aforementioned steps, server has been listening to any clients for the connection. The backlog integer is used to specify the maximum length of the incoming connection queue.
_backlog is adjusted to be within the range from 200 to 65535 (connections). If a client attempts to connect to a server whose backlog queue is full, it will receive a connection refused error.
_If no error occurs, listen returns **zero**. Otherwise, a value of **SOCKET_ERROR** is returned.

- ***Accept Connections***

When server received a connection request from client, it needs to accept to service the client.

```
SOCKET WSAAPI accept(
  SOCKET   s,
  sockaddr *addr,
  int      *addrlen
);
```

- ***Sending and Receiving***

  _Now we are at the point where our server has accepted a client's connection request and created anew socket for it to accept and send messages.

```
int WSAAPI send(
  SOCKET     s,
  const char *buf,
  int        len,
  int        flags
);
```

```
int WSAAPI recv(
  SOCKET s,
  char   *buf,
  int    len,
  int    flags
);
```

  _The two functions are very similar. A socket is required to either send to or receive from and then data in the buffer is either sent or data is received into an empty buffer.

  _The usage of flag is to peek at the packet of the queue without actually removing it from the queue.

- ***Tidying up***

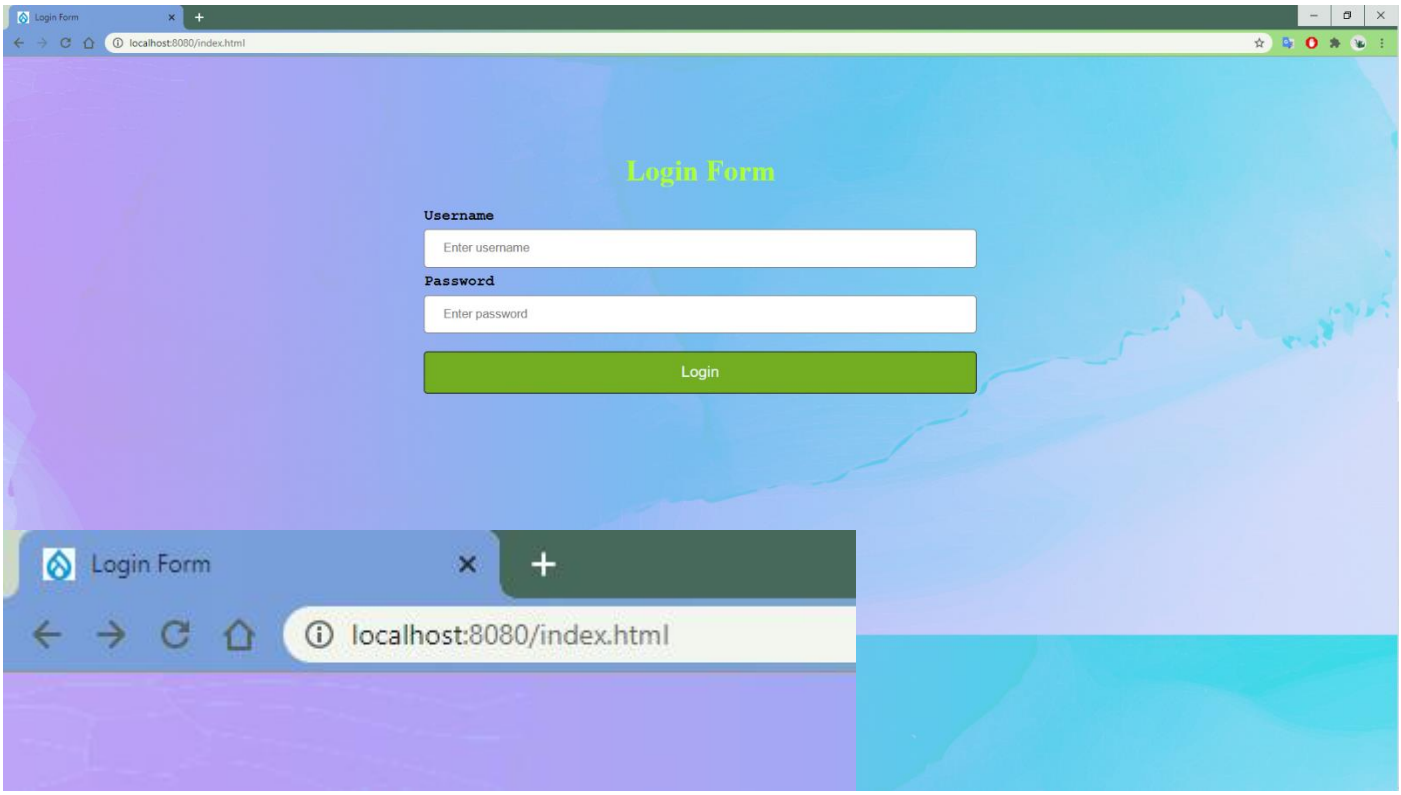
  When we are finished with a socket, we need to close it.

```
int WSAAPI closesocket(
  SOCKET s
);
```

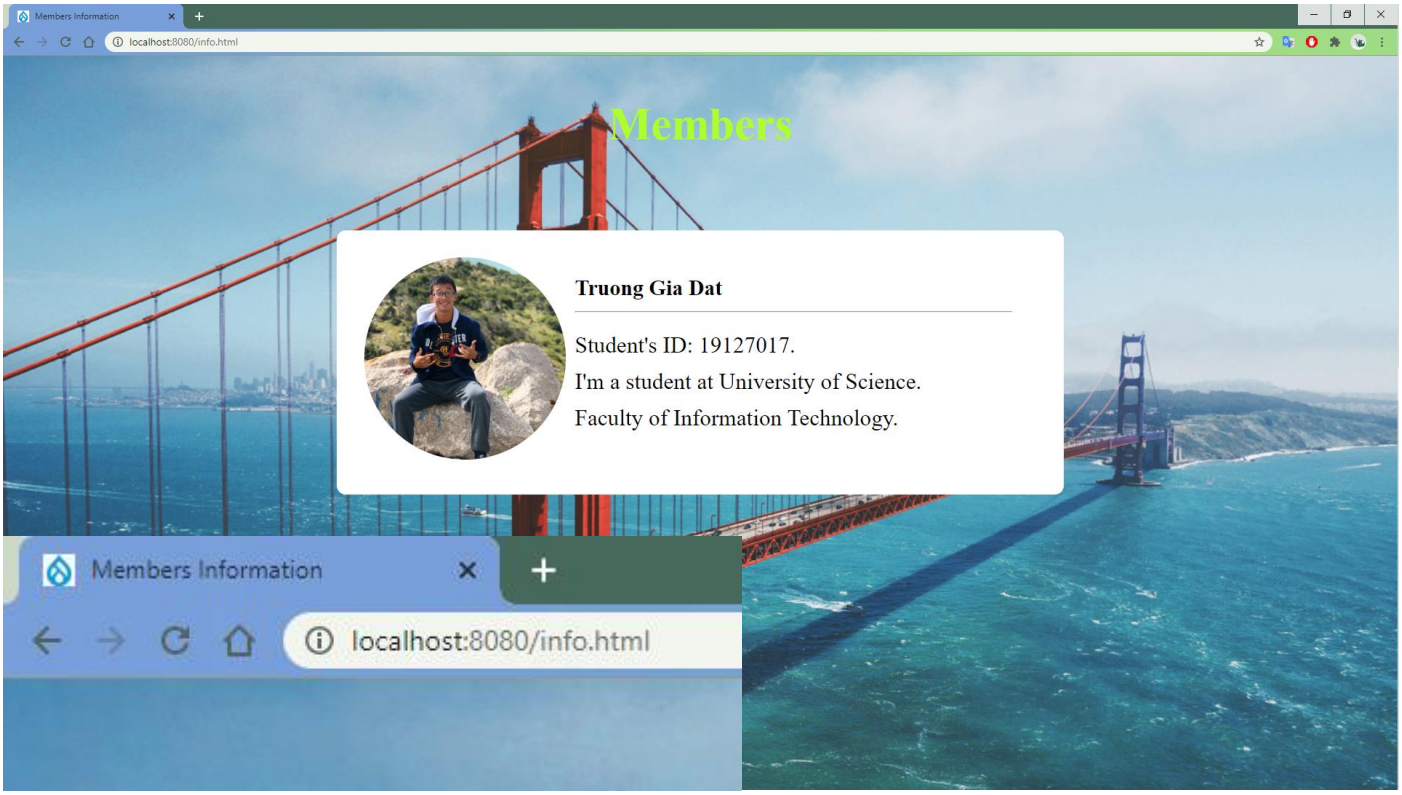
  _We provide the socket descriptor of the socket that we want to close. This ensures that the socket is closed cleanly and the memory it has used is reclaimed.

Program Demonstration

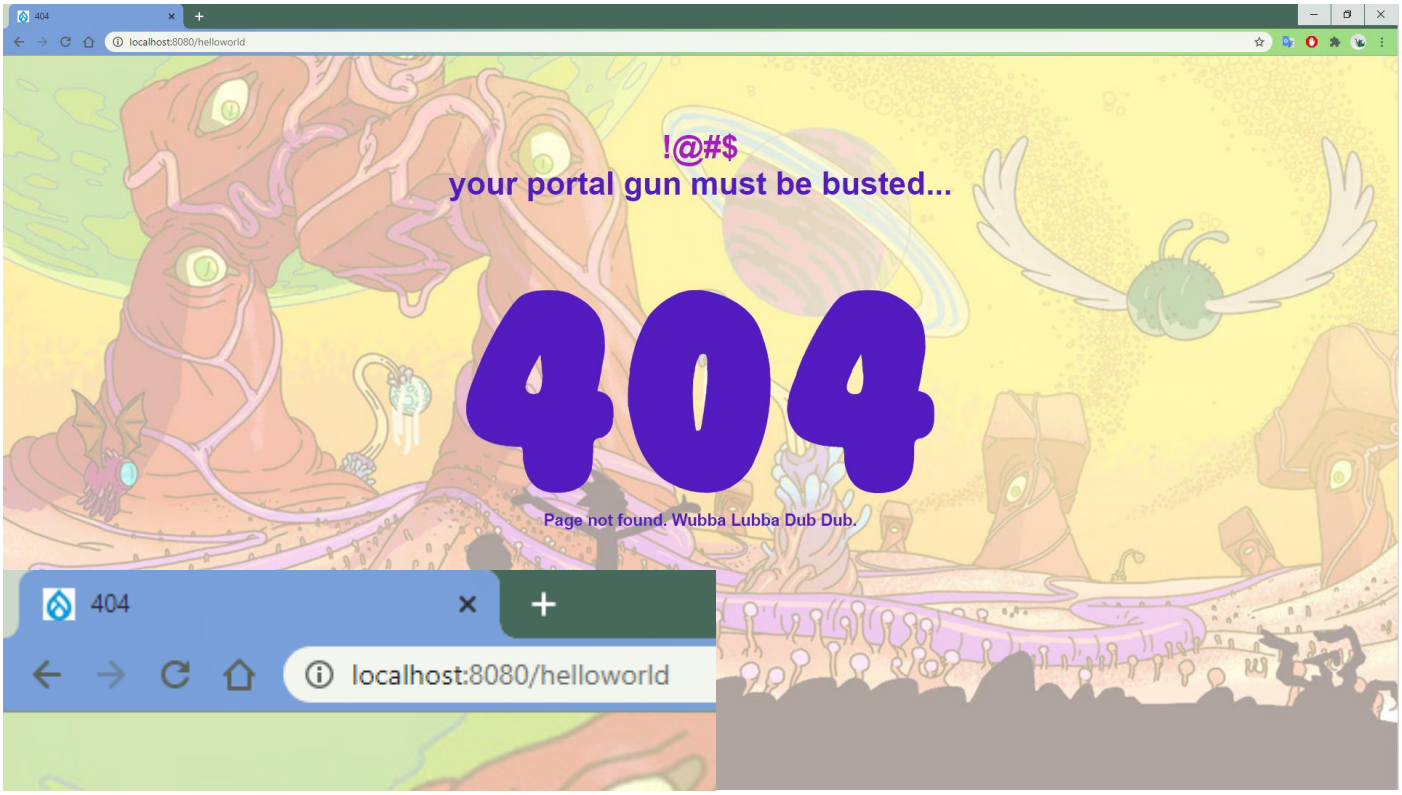
We run the project that has been built and then open a web browser. Type localhost:[port] or 127.0.0.1:[port] (port is optionally chosen in project by developer). Web browser will open index.html file (Dir: localhost:[port]/index.html). (port in this case is **8080**)



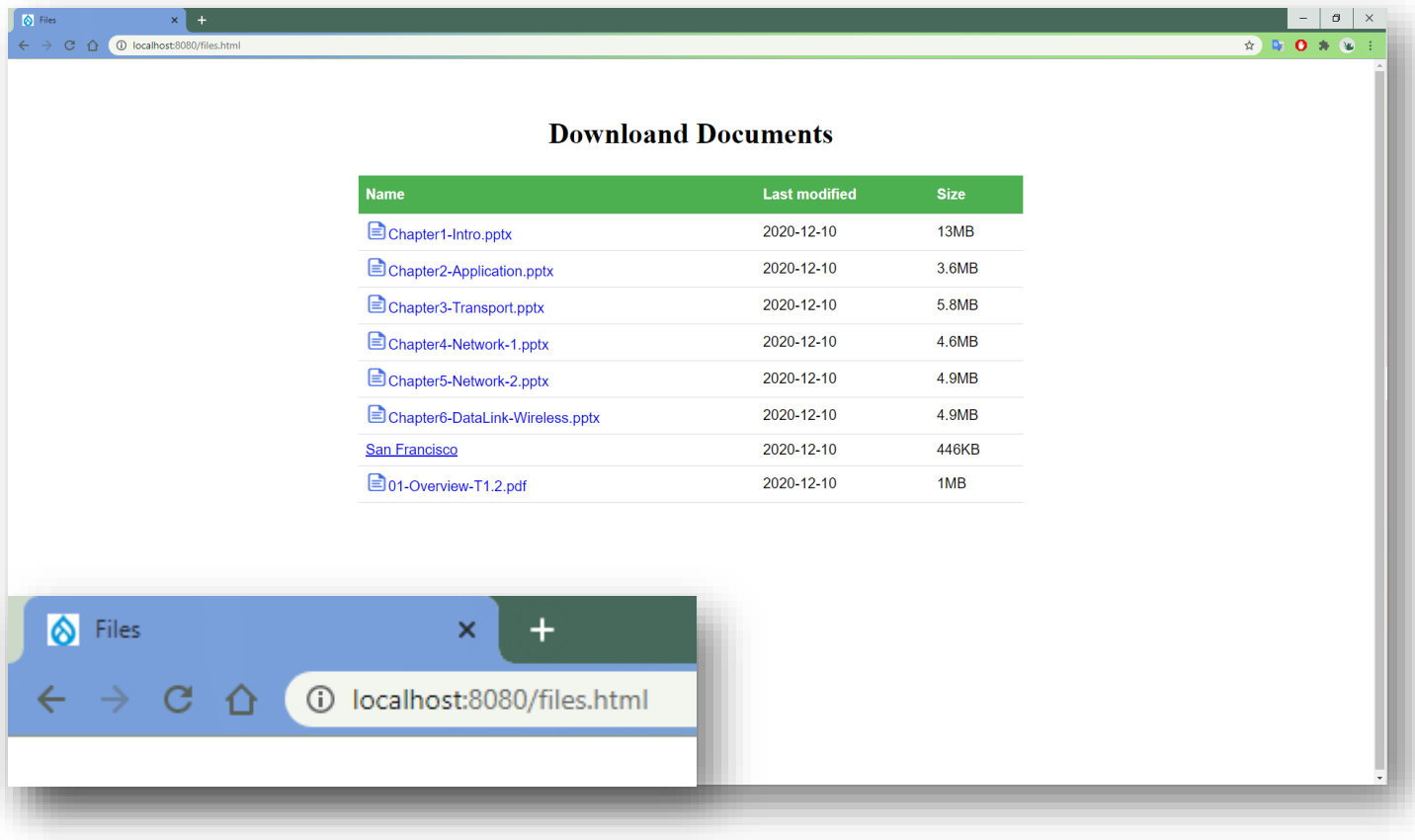
Enter username and password. If values of the input are correct, web browser will instantly navigate to info.html file. Otherwise, 404.html file will be displayed.



If client types in an unknown address (example: localhost:[port]/helloworld), web browser will instantly open 404.html.



_When client types in localhost:[port]/files.html, web browser will move to a page in which client is authorized to download files.



NOTES:

- Web browser:
  - Working stably and effectively in **Microsoft Edge** and **Google Chrome**.
- Features:
  - Server currently services to a single client, respectively, which means 1 – 1.
  - In processing to enhance server be able to interact with multiple clients.

REFERENCES

1. Microsoft Documents: https://docs.microsoft.com/en-us/windows/win32/winsock/creating-a-basic-winsock-application.
2. Features in `WinSock2.h` header file: https://docs.microsoft.com/en-us/windows/win32/api/winsock2/.
3. HTTP fundamentals: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.
4. HTTP status code: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.