ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN SOCKET

MÔN: MẠNG MÁY TÍNH

| Sinh Viên Thực Hiện |

Trần Hữu Chí Bảo - 18120288

Cao Tất Cường - 18120296

Nguyễn Bách Tùng - 18120641

Thành phố Hồ Chí Minh – 2020

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN SOCKET

MÔN: MẠNG MÁY TÍNH

| Giáo Viên Hướng Dẫn |

ThS. Lê Hà Minh

Thành phố Hồ Chí Minh – 2020

MỤC LỤC

PHÂN CÔNG VÀ ĐÁNH GIÁ

I. Phân Công và Đánh giá:

Họ tên	Phân Công	Hoàn thành
Trần Hữu Chí Bảo	Tạo trang 404.html và trang info.html và viết báo cáo.	100%
Cao Tất Cường	Viết code socket và thực thi chương trình chạy thành công.	100%
Nguyễn Bách tùng	Tạo trang index.html và viết báo cáo.	100%

- Đã hoàn thiện hết các yêu cầu được giao.
- Mức độ hoàn thành: 100%.

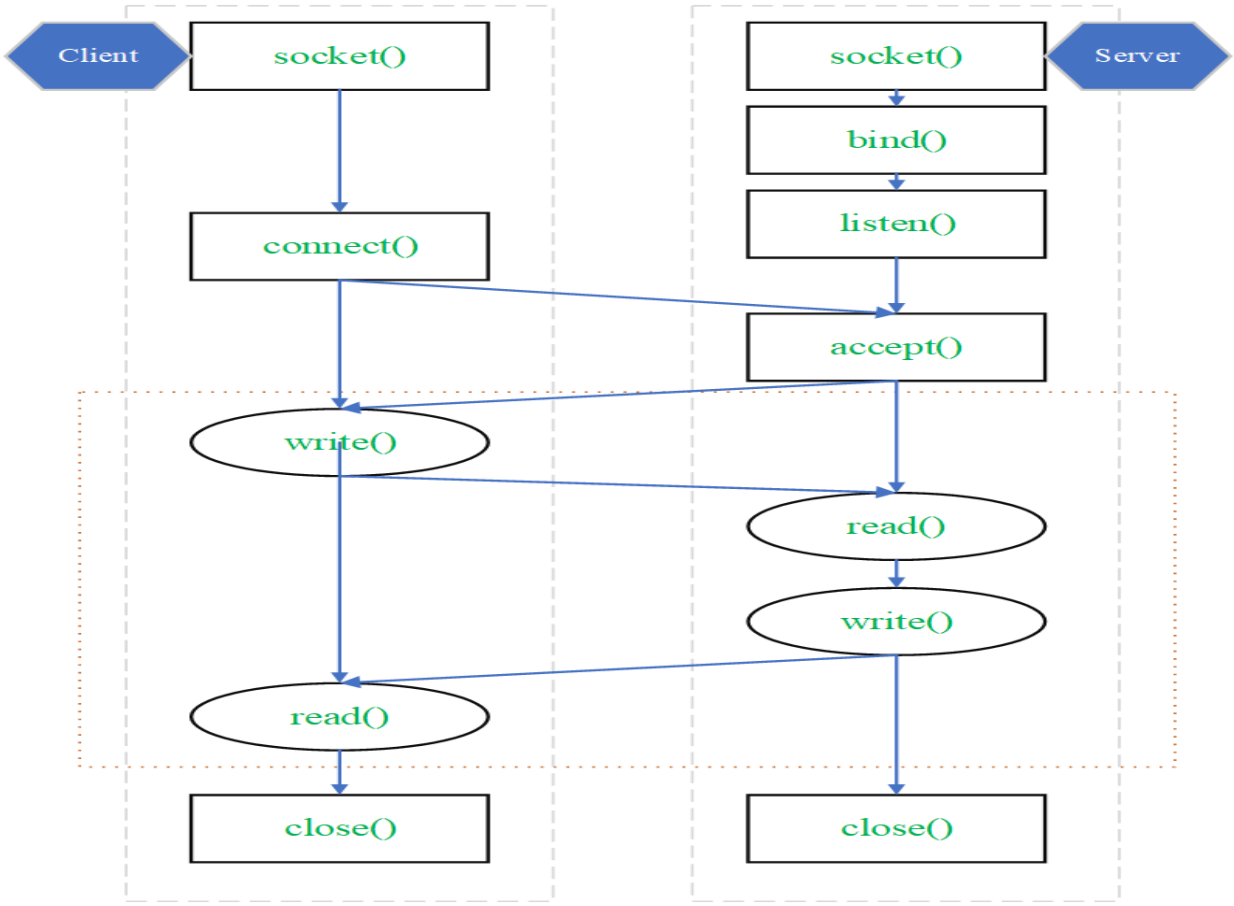
II. Link github:

https://github.com/cuong091200/MMT-WEB

MÔ TẢ CÁCH THỰC HIỆN ỨNG DỤNG

I. Nguyên lý hoạt động:

- **Ngôn ngữ sử dụng**: Java
- **Nguyên lý:**



1. Server tạo socket, gán số hiệu cổng (bind) và lắng nghe (listen) yêu cầu kết nối. Server sẵn sàng phục vụ Client.socket():
   - Bind(): gán số hiệu cổng.
   - Listen(): lắng nghe yêu cầu kết nối từ Client.
2. Client tạo socket, yêu cầu thiết lập kết nối với server:
   - Connect(): client gửi yêu cầu nối đến server có IP và Port xác định.
   - Accept(): server chấp nhận kết nối, một kênh giao tiếp ảo được hình thành, Client và Server trao đổi giao tiếp thông qua kênh này.

3. Trao đổi thông tin:
    - Sau khi xác lập kết nối, server thực hiện lệnh read() và nghẽn cho đến khi nhận được yêu cầu từ phía Client.
    - Sau khi đọc và thực thi yêu cầu, server sẽ gởi kết quả về Client bằng lệnh write()
    - Sau khi gửi, server tiếp tục vòng lặp chờ nhận request từ Client bằng read()
4. Kết thúc phiên làm việc:
    - Câu lệnh read write lặp lại nhiều lần.
    - Kênh ảo sẽ bị xóa khi client hoặc servre thực hiện lệnh close().

II. Source Code thực thi chương trình:

- Tạo một hệ thống socket để nhận kết nối trên một cổng TCP nhất định. Để ví dụ ta chọn port 80. Sử dụng lớp ServerSocket của Java để tạo ra server nhận yêu cầu. Xem hình ảnh bên dưới:

```
//Tao ra 1 socketserver để lắng nghe kết nối
public void StartServer() {
    ServerSocket server = null;
    try {
        // Mở 1 server lắng nghe ở port 80
        server = new ServerSocket(PORT);
        System.out.println("Waiting for Clients ");
        while (true) {
            //Mở 1 socket riêng để chấp nhận 1 kết nối đến server
            Socket socket = server.accept();
            //Địa chỉ IP của cilent kết nối đến server
            System.out.println("Client Connected From : " + socket.getLocalAddress().toString());
            Handler handler = new Handler(socket);
            handler.run();
            socket.close();
        }

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            server.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Khi vào vòng lặp thì server sẽ chờ client kết nối đến và accept. Ngay lúc đó nó sẽ trả về một Socket object để đọc và xử lý yêu cầu từ phía Client.

- Khi kết nối tới http://127.0.0.1/index.html , trình duyệt sẽ gửi request tới cho server. Server đọc yêu cầu này, sử dụng InputStream để đọc từ Client socket (tốt nhất là dùng BufferedReader) và OutputStream để trả về dữ liệu cho Client socket.

```
public class Handler implements Runnable {
    private Socket socket;
    private PrintWriter out;
    private BufferedReader reader;
    private String message;

    public Handler() {
    }

    public Handler(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        try {
            //Đọc các byte dữ liệu từ client gửi về và giải mã chúng thành các ký tự
            InputStreamReader inputStreamReader = new InputStreamReader(socket.getInputStream());

            //chuyển sang BufferedReader để đọc hiệu quả hơn
            reader = new BufferedReader(inputStreamReader);

            //Trả về đầu ra cho socket
            OutputStream outputStream = socket.getOutputStream();
            //Đọc các lý tự server phản hồi về client và mã hóa thành byte

            out = new PrintWriter(outputStream);
            //Đọc từng dòng dữ liệu gửi từ client về server
            message = reader.readLine();
```

- Tin nhắn gửi từ client có dạng [Method] [/filepath] HTTP/1.1, method có thể là 2 dạng GET hoặc POST.
- Nếu là GET, ta kiểm tra liệu URL có tồn tại không. Nếu không, chuyển hướng về 404.html.

```
// Nếu đường dẫn client nhập vào là địa chỉ ip thì chuyển đến trang 404.html
if (filePath.equals("/") || filePath.isEmpty())
    filePath = "/404.html";

// Xử lý phương thức GET
if (method.equals(HTTPServer.METHODS[0])) {

    System.out.println("requesting: " + filePath);

    // Nếu url không tồn tại thì chuyển đến trang 404.html
    if (!(new File(HTTPServer.FOLDERHTML + filePath)).exists()) {
        String response = HandleMethodGet(false, filePath);
        //Xuất dữ liệu mới đọc được ra lên client(code của 404.html)
        PrintWriter pw = new PrintWriter(out);
        pw.print(response);
        pw.close();
    }
```

- Tiếp đến xử lý file html và ảnh

```
// Nếu url tồn tại thì chuyển đến trang request
if (HTTPServer.TEXT.equals(filePath.split("[.]")[1])) {
    String response = HandleMethodGet(true, filePath);
    //Xuất dữ liệu mới đọc được ra lên client(code file request)
    PrintWriter pw = new PrintWriter(out);
    pw.print(response);
    pw.close();
}

// Xử lý ảnh .png
else if (HTTPServer.IMAGES.equals(filePath.split("[.]")[1])) {
    FileInputStream is = new FileInputStream(HTTPServer.FOLDERHTML + filePath);
    OutputStream os = socket.getOutputStream();
    int s;
    // Lấy dữ liệu của ảnh
    while ((s = is.read()) > -1) {
        os.write(s);
    }
    os.flush();
    is.close();
    os.close();
}
```

```
private String HandleMethodGet(boolean temp, String filepath) throws IOException {
    //Nếu temp = false (url khong tồn tai) thì chuyển hướng đến 404.html
    if (!temp) {
        return "<head><meta http-equiv='Refresh' content='0; URL=/404.html'></head>";
    }
    //Lấy đường dẫn đi vào file hrml tương ứng của request
    File returnFile = new File(HTTPServer.FOLDERHTML + filepath);
    // Đọc code html lưu vào trong respone để gửi lên client
    FileInputStream fileInputStream = new FileInputStream(returnFile);

    String response = "";
    int s;
    while ((s = fileInputStream.read()) != -1) {
        response += (char) s;
    }
    fileInputStream.close();
    return response;
}
```

- Nếu là POST, đọc dữ liệu đã bị mã hóa, chuyển về UTF – 8.

```
// Phương thức POST
else if (HTTPServer.METHODS[1].equals(method)) {
    while (!(message = reader.readLine()).equals(""))
        System.out.println(message);
        ;

    // Đọc dữ liệu đã bị mã hóa từ method post
    StringBuilder builder = new StringBuilder();
    while (reader.ready())
        builder.append((char) reader.read());

    // Chuyển dữ liệu bị mã hóa sang UTF-8
    String data = URLDecoder.decode(builder.toString(), "UTF-8");
    System.out.println(data);
    String response = HandleMethodPost(data);
    out.write(response);

}
```

```
private String HandleMethodPost(String data) {
    String response = "";
    int count = 0;
    String[] pairs = data.split("&");
    //Kiểm tra tài khoản mật khẩu client nhập vào
    for (String pair : pairs) {
        if (pair.equals("username=admin") || pair.equals("pass=admin")) {
            count++;
        }
    }
    //Nếu đúng chuyển hướng sang trang info.html nếu sai thì 404.html
    if (count == 2) {
        response += "<head><meta http-equiv='Refresh' content='0; URL=/info.html'></head>";
    } else {
        response += "<head><meta http-equiv='Refresh' content='0; URL=/404.html'></head>";
    }
    return response;
}
```

- Chương trình Server được viết theo kiểu phục vụ tuần tự, phục vụ cho 1 client tại một thời điểm, một cổng duy nhất (trong trường hợp này ta sử dụng port 80).

III. Giao diện các trang web:

3.1 Trang 404.html:

- Tạo thẻ div bao quanh toàn bộ khung chữ và nút button Go back Home. Sau đó, định dạng thẻ div bao gồm định dạng chiều cao, chiều rộng, căn lề, …
- Tiếp theo định dạng chữ OOP! Và 404 – The Page can't be found vào trong 1 thẻ div.
- Và định dạng dòng chữ Go To HomPage và gắn vào đó link đến cái trang muốn chuyển tới khi nháy chuốt vào dòng chữ đó.

```
@media only screen and (max-width: 480px) {
  .notfound .notfound-404 {
    height: 148px;
    margin: 0px auto 10px;
  }
  .notfound .notfound-404 h1 {
    font-size: 86px;
  }
  .notfound .notfound-404 h2 {
    font-size: 16px;
  }
  .notfound a {
    padding: 7px 15px;
    font-size: 14px;
  }
}
</style>
<body>

<div id="notfound">
  <div class="notfound">
    <div class="notfound-404">
      <h1>Oops!</h1>
      <h2>404 - The Page can't be found</h2>
    </div>
    <a href="index.html">Go TO Homepage</a>
  </div>
</div>

</body>
</html>
```

```
#notfound {
  position: relative;
  height: 100vh;
}

#notfound .notfound {
  position: absolute;
  left: 50%;
  top: 50%;
  -webkit-transform: translate(-50%, -50%);
      -ms-transform: translate(-50%, -50%);
          transform: translate(-50%, -50%);
}

.notfound {
  max-width: 520px;
  width: 100%;
  line-height: 1.4;
  text-align: center;
}

.notfound .notfound-404 {
  position: relative;
  height: 200px;
  margin: 0px auto 20px;
  z-index: -1;
}

.notfound .notfound-404 h1 {
  font-family: 'Montserrat', sans-serif;
  font-size: 236px;
  font-weight: 200;
  margin: 0px;
  color: #211b19;
  text-transform: uppercase;
  position: absolute;
  left: 50%;
```

➔ Kết quả của trang 404 Not Found

3.2 Trang index.html

- Tạo thẻ div bao quanh toàn bộ khung màu trắng.
  Sau đó, định dạng thẻ div bao gồm định dạng chiều cao, chiều rộng, căn lề, …
- Thêm hình ảnh biểu tượng vào trong trang.
- Tiếp theo dùng thẻ input để thêm thông tin user và password.
- Và định dạng dòng chữ Login và gắn vào đó link đến cái trang muốn chuyển tới khi nháy chuốt vào dòng chữ đó.

```
<body>
    <div class="wrapper fadeInDown">
        <div id="formContent">

          <h2 class="active"> Sign In </h2>

          <div class="fadeIn first">
            <img src="login-img.png" id="icon" alt="User Icon" />
          </div>

          <form method="post" action="">
            <input type="text" id="login" class="fadeIn second" name="username" placeholder="login">
            <input type="password" id="password" class="fadeIn third" name="pass" placeholder="password">
            <input type="submit" class="fadeIn fourth" value="Log In">
          </form>
        </div>
      </div>
</body>
</html>
```

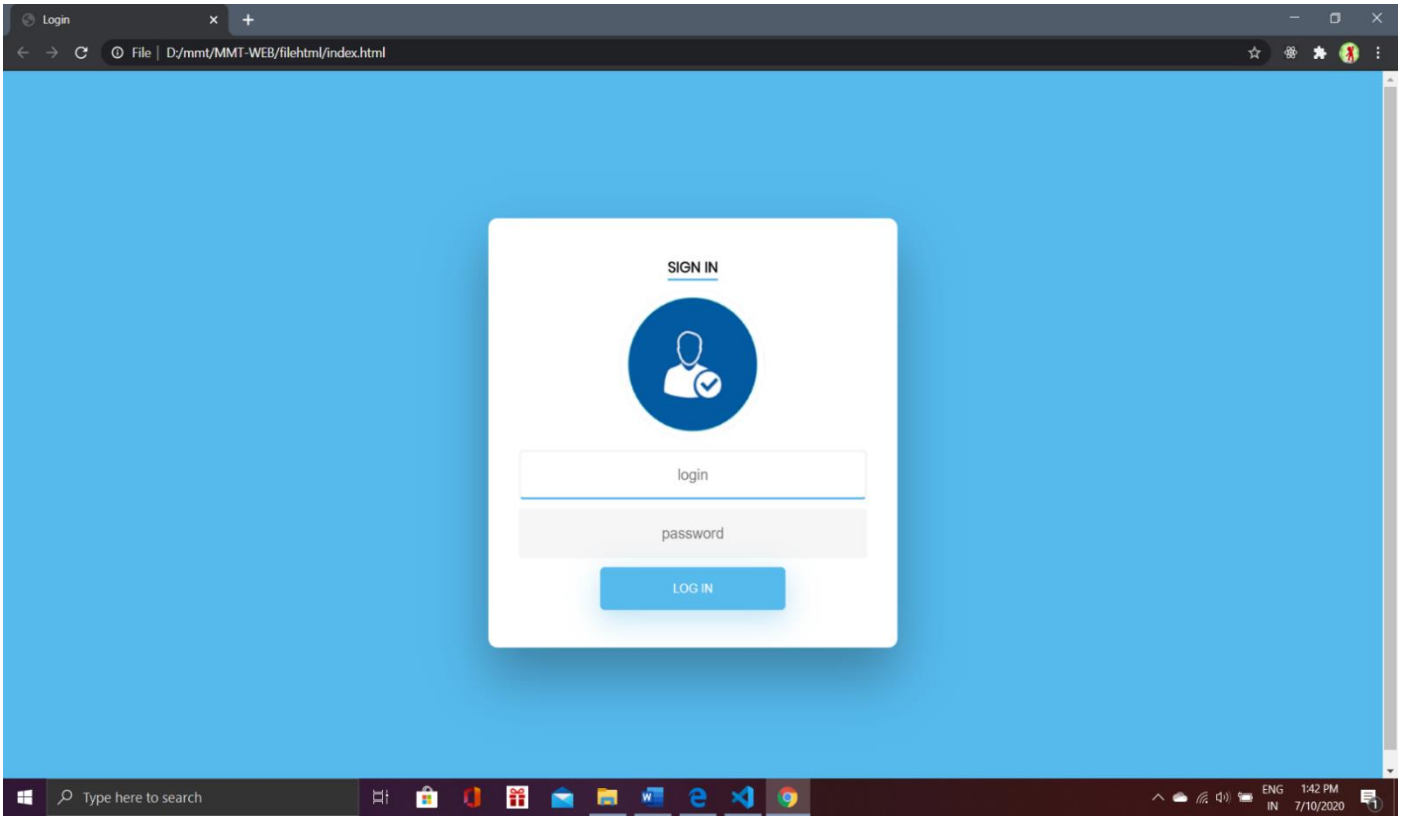
```
#formContent {
-webkit-border-radius: 10px 10px 10px 10px;
border-radius: 10px 10px 10px 10px;
background: #fff;
padding: 30px;
width: 90%;
max-width: 450px;
position: relative;
padding: 0px;
-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);
box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);
text-align: center;
}

h2.inactive {
color: #cccccc;
}

h2.active {
color: #0d0d0d;
border-bottom: 2px solid #5fbae9;
}

input[type=button], input[type=submit], input[type=reset]  {
background-color: #56baed;
border: none;
color: white;
padding: 15px 80px;
text-align: center;
text-decoration: none;
display: inline-block;
text-transform: uppercase;
font-size: 13px;
-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);
box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);
-webkit-border-radius: 5px 5px 5px 5px;
border-radius: 5px 5px 5px 5px;
```

➔ Kết quả của trang Index.html



3.3 Trang Info.html

- Tạo thẻ div bao quanh toàn bộ khung. Sau đó, định dạng thẻ div bao gồm định dạng chiều cao, chiều rộng, căn lề, …
- Định dạng dòng chữ Meet out team (căn lề, thêm hiệu ứng).
- Định dạng 1 thẻ div bao gồm thêm hình ảnh, thêm font awesome vào truyền link liên kết đến khi nháy chuột vào biểu tượng.
- Định dạng tương tự cho 2 thẻ div tiếp theo.

```
body{
  margin: 0;
  padding: 0;
  font-family: "montserrat";
}
.team-section{
  background: #f1f1f1;
  text-align: center;
}
.inner-width{
  max-width: 1200px;
  margin: auto;
  padding: 40px;
  color: #333;
  overflow: hidden;
}
.team-section h1{
  font-size: 20px;
  text-transform: uppercase;
  display: inline-block;
  border-bottom: 4px solid;
  padding-bottom: 10px;
}
.pe{
  float: left;
  width: calc(100% / 3);
  overflow: hidden;
  padding: 40px 0;
  transition: 0.4s;
}
.pe:hover{
  background: #ddd;
}
.pe img{
  width: 120px;
  height: 120px;
}
```

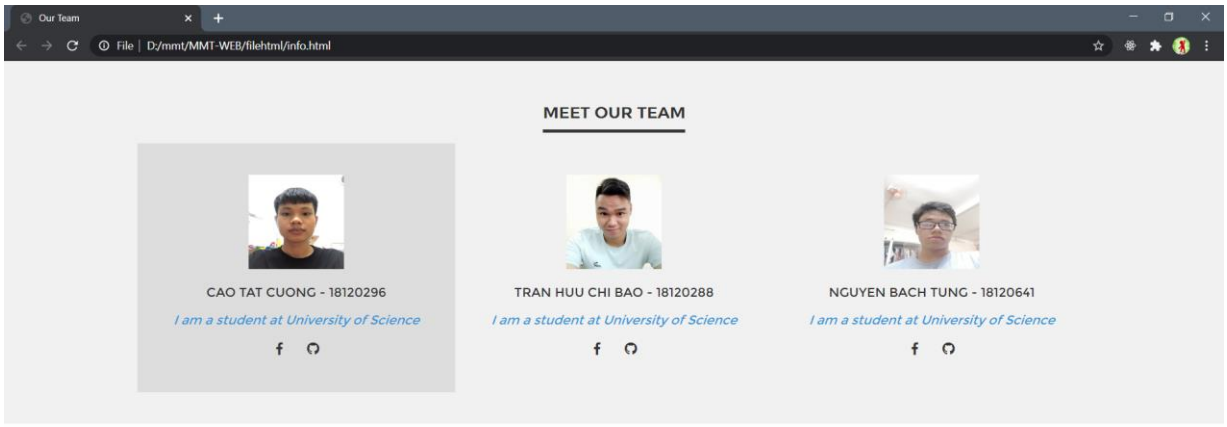
```html
<div class="team-section">
  <div class="inner-width">
    <h1>Meet Our Team</h1>
    <div class="pers">

      <div class="pe">
        <img src="p1.png" alt="">
        <div class="p-name">Full Name</div>
        <div class="p-des">Designer</div>
        <div class="p-sm">
          <a href="#"><i class="fab fa-facebook-f"></i></a>
          <a href="#"><i class="fab fa-github"></i></a>
        </div>
      </div>

      <div class="pe">
        <img src="p2.png" alt="">
        <div class="p-name">Full Name</div>
        <div class="p-des">Designer</div>
        <div class="p-sm">
          <a href="#"><i class="fab fa-facebook-f"></i></a>
          <a href="#"><i class="fab fa-github"></i></a>
        </div>
      </div>

      <div class="pe">
        <img src="p3.png" alt="">
        <div class="p-name">Full Name</div>
        <div class="p-des">Developer</div>
        <div class="p-sm">
          <a href="#"><i class="fab fa-facebook-f"></i></a>
          <a href="#"><i class="fab fa-github"></i></a>
        </div>
      </div>
    </div>
  </div>
</div>
```

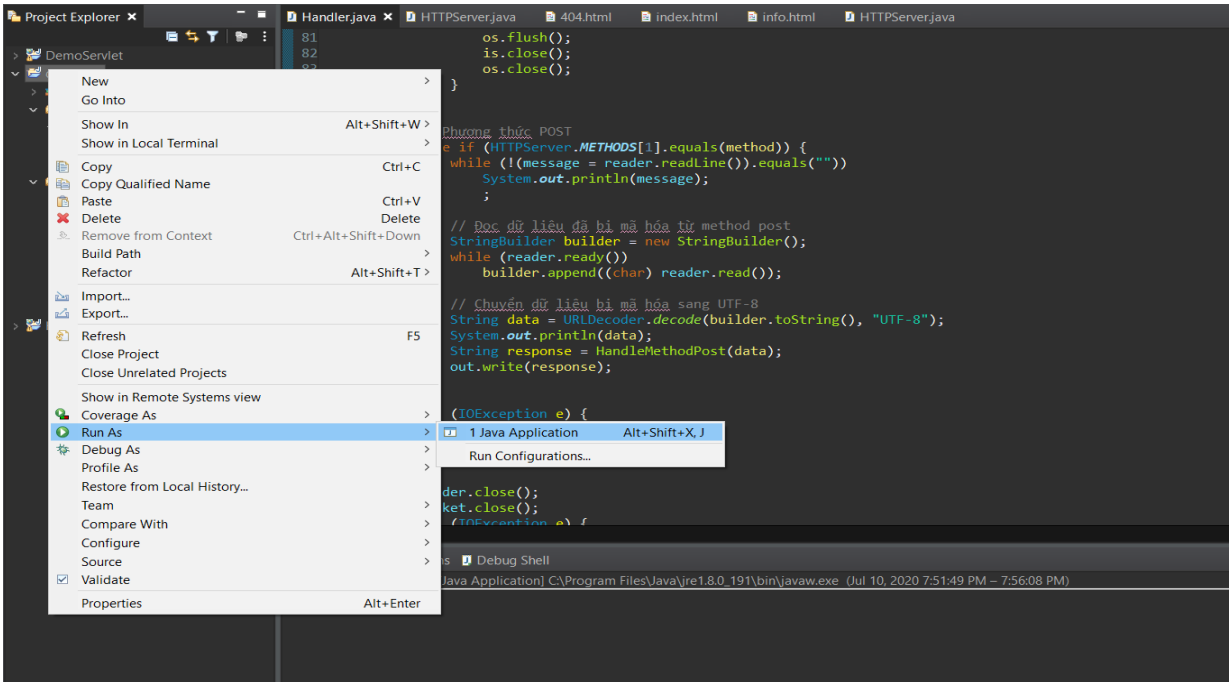
→ Kết quả của trang Info.html



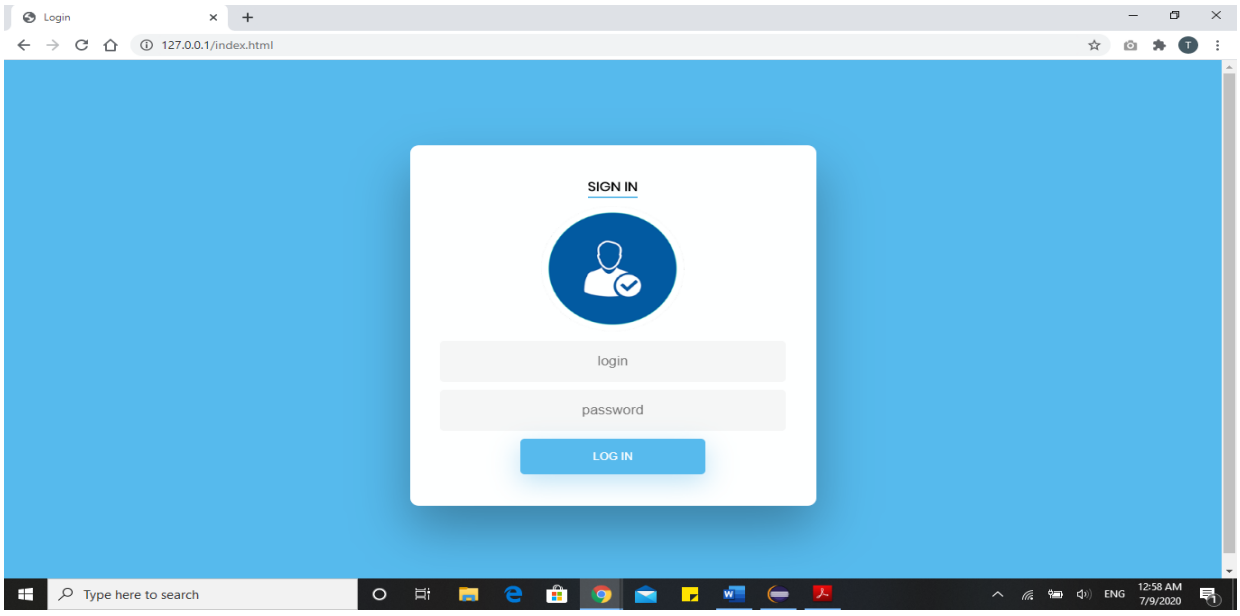
TEST THỬ ỨNG DỤNG

- Trên Server:

  - Nhấp chuột phải Project → RunAs → 1. Java Application

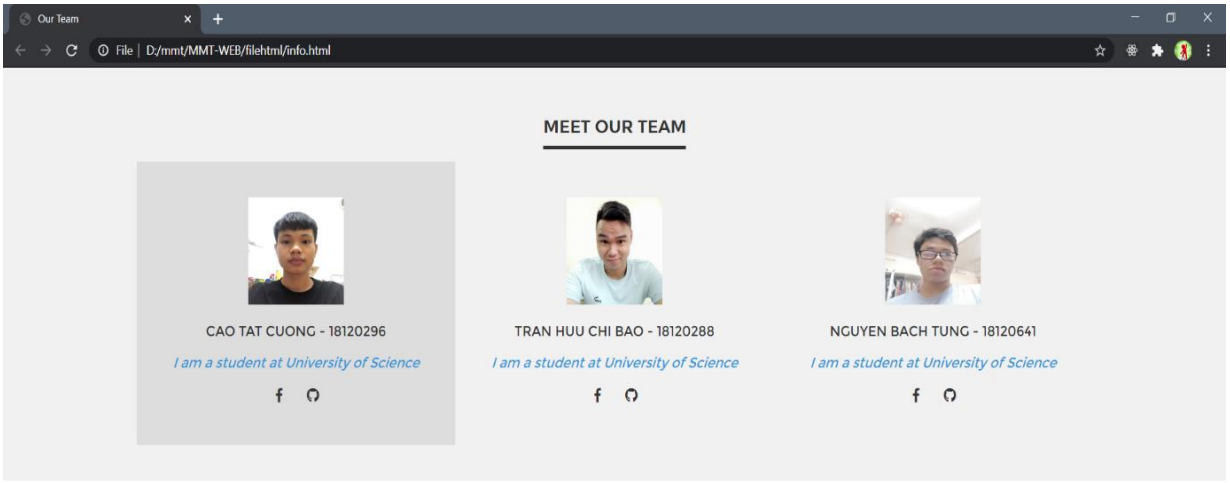


  - Chạy client:
  - Từ web browser:
  - Nhập đường dẫn 127.0.0.1/index.html.

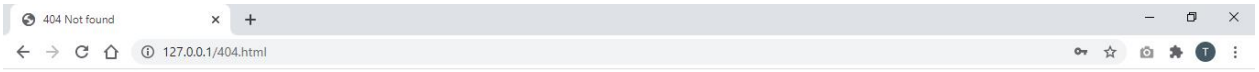


(trang log in)

- Nếu nhập user: admin và password: admin thì sẽ chuyển đến trang info.html. Ngược lại, nếu người dùng nhập không đúng 1 trong 2 cái trên (user hoặc password) thì sẽ chuyển đến trang 404.html.



(trang info)



(404 Error)

TÀI LIỆU THAM KHẢO

https://www.w3schools.com/html

https://colorlib.com/wp/cat/

https://developer.mozilla.org/vi/docs/Web/API/WebSockets_API/Writing_a_WebSocket_server_in_Java