**Bài 1 :  Làm quen với SocketIO**

1. Tạo folder **SERVER** để lưu thông tin project
2. Tạo Project: **npm init**
3. Download các thư viện express, ejs và socketio về: **npm install express ejs socket.io**
4. Tạo file **index.js** để tạo server express trong folder này
5. Tạo thư mục **views** để chứa source code từ client: **trangchu.ejs**
6. Cài đặt thêm **socketio** vào file **index.js** rồi sau đó chỉnh sửa file **index.js** để lắng nghe xem có yêu cầu kết nối từ client hay không?
7. Chỉnh sửa file **trangchu.ejs** để thực hiện kết nối với server
8. Chỉnh sửa file **index.js** để lắng nghe xem có yêu cầu kết nối/ngắt kết nối từ client hay không?

**Bài 2 : Socket io cơ bản**

1. Chỉnh sửa file **trangchu.ejs** để test xem nếu kích vào nút thì nó có bắt được sự kiện hay không?
2. Thực hiện gửi dữ liệu từ client lên server (**emit**), và server nhận dữ liệu từ client gửi lên (**on**)
3. Server thực hiện gửi dữ liệu trả về cho tất cả các client thông qua phương thức **io.sockets.emit**
4. Server thực hiện gửi dữ liệu trả về cho client vừa emit thông qua phương thức **socket.emit**
5. Server thực hiện gửi dữ liệu trả về cho các client, trừ client vừa emit thông qua phương thức **socket.broadcast.emit**
6. Bài Tập 1
   1. Khách hàng gửi hai giá trị a, b lên server bằng phương thức emit
   2. Server lắng nghe dữ liệu từ khách hàng gửi lên, sau đó tính tổng a+b, gửi kết quả về cho client vừa gửi yêu cầu
7. Bài Tập 1B
   1. Khách hàng gửi ba giá trị a, b, c lên server bằng phương thức emit (gửi mảng giá trị)
   2. Server lắng nghe dữ liệu từ khách hàng gửi lên, sau đó tính tổng a+b+c, gửi kết quả về cho client vừa gửi yêu cầu (gửi mảng giá trị)
8. Bài tập 2
   1. Tạo các nút Đỏ, Xanh, Vàng ở trang của khách hàng
   2. Khi người dùng bấm vào nút màu nào thì đổi màu nền của tất cả các trang khác
9. Bài Tập 3: (Tự luyện) Xây dựng ứng dụng chat online đơn giản

**Bài 3: Xây dựng web chat online**

**I – Mô tả**

1. Người dùng đăng ký tên để chat: Người dùng sẽ gửi username lên server, nếu username đã tồn tại thì server thông báo đăng ký thất bại cho người dùng này biết, nếu đăng ký thành công thì thông báo cho người dùng này biết, đồng thời cũng thông báo cho các người dùng khác biết có một người mới đăng ký thành công
2. Nếu đăng ký thành công thì chuyển sang màn hình chat, gồm cột bên trái hiện thị tất cả những người đang online, cột bên phải hiện thị khung để chat, nếu người dùng nhập vào nội dung chat và bấm gửi thì nội dung chat sẽ được hiển thị lên máy của tất cả các người dùng
3. Nếu có một người dùng nào đó thoát ra thì xóa tên người đó khỏi danh sách những người đang online ở cột bên trái
4. Nếu có người đang nhập văn bản thì hiện thị ra thông báo để các người dùng khác biết

**II – Các bước thực hiện**

1. Xây dựng giao diện cho phần mềm
2. Thực hiện client gửi username lên server, server nhận kết quả gửi lên từ client, sau đó kiểm tra:
   1. Nếu username đã tồn tại thì thông báo về client đăng ký thất bại
   2. Nếu đăng ký thành công thì ẩn form đăng ký đi, đồng thời hiện form chat ra với user vừa gửi form đăng ký
   3. Cập nhật lại danh sách tất cả các user đang online trên trang của tất cả các người dùng
   4. Nếu có người dùng thoát ra: hiện thị lại form login cho người ấy đồng thời xóa người này ra khỏi danh sách đang online
   5. Nếu có người dùng mất kết nối với server: xóa người này ra khỏi danh sách đang online
3. Chức năng chat
   1. Khi người dùng nhập nội dung vào khung chat và nhấn nút SEND: lấy nội dung trong khung chat và emit lên server
   2. Sever nhận dữ liệu từ client gửi lên, sau đó xác định username của client vừa emit thông tin lên, sau đó gửi trả lại cho các client gồm username và nội dung chat
   3. Client nhận thông tin từ server emit về và hiển thị ra khung chat
   4. Nếu có ai đó đang nhập đoạn chat thì hiển thị cho các user khác được biết

**Bài 4: Room & Namespace**

**I – Lý thuyết**

1. Không có hàm tạo ROOM mà mỗi khi một client truy cập vào localhost, server sẽ tạo một socket cho client và được chứa trong 1 ROOM
2. Một ROOM có thể chứa nhiều socket, nếu muốn socket vào trong ROOM thì sử dụng phương thức join vào ROOM
3. Một socket có thể được chứa trong nhiều ROOM

**II- Các bước thực hiện**

* **Chức năng cập nhật danh sách phòng và phòng hiện tại**

1. Ban đầu, khi một client truy cập localhost thì server tạo ra 1 ROOM để chứa nó, có tên ROOM trùng với socket.id
2. Khi client nhập tên ROOM và nhấn nút tạo ROOM, client sẽ gửi tên ROOM lên server, server sẽ nhận tên ROOM, sau đó đưa socket đó vào trong ROOM vừa tạo, đồng thời tạo biến **socket.Phong** là tên ROOM vừa gửi lên
3. Server tạo mảng để chứa tất cả các ROOM hiện có, sau đó emit mảng này về cho các client , đồng thời cũng emit tên ROOM về cho client vừa gửi tên ROOM lên!
4. Client nhận mảng các ROOM và hiển thị ra cột danh sách các phòng (bên trái trình duyệt), đồng thời cũng hiện thị phòng hiện tại của client đó (bên phải trình duyệt)

* **Chức năng chat trong cùng một phòng**

1. Khi nhập nội dung và nhấn nút chat, client gửi lên server nội dung chat
2. Server lắng nghe nội dung từ client gửi lên và gửi về tất cả các client trong phòng đó theo phương thức **io.sockets.in(socket.Phong).emit**
3. Client lắng nghe nội dung chat từ server gửi về và hiển thị ra trình duyệt

**Bài 5: Đưa ứng dụng lên Heroku & event cuối khoá**

1. **Tạo Project**

**I - Mô tả**

1. Tạo 1 project để nhập thông tin gồm họ tên, email, số điện thoại
2. Gửi các thông tin này lên server, server nhận dữ liệu rồi gửi danh sách những người đã đăng ký về client
3. Client nhận danh sách từ client gửi về, hiển thị danh sách ra trình duyệt
4. Đưa dự án lên heroku

**II – Các bước thực hiện**

1. Tạo giao diện cho project
2. Tạo server express
3. Tạo trang truy cập từ client
4. Client nhập thông tin và gửi dữ liệu bằng phương thức **socket.emit()**
5. Server lắng nghe và nhận thông tin từ client gửi lên bằng phương thức **socket.on()**
6. Tạo mảng chứa các thông tin này và push các thông tin từ client gửi lên vào mảng
7. Server gửi danh sách các học viên về cho client bằng phương thức **io.sockets.emit()**
8. Client lắng nghe danh sách từ server gửi về và hiển thị ra trình duyệt
9. **Đưa project lên Heroku**
10. Đăng ký tài khoản tại trang <https://www.heroku.com/> và đăng nhập vào
11. Tạo một app mới để chứa project
12. Deploy ứng dụng NodeJS lên Heroku từ GitHub
13. Chạy ứng dụng