Bài tập: WCF Chat.

1. Tạo project Windows Store: Chọn New Project -> Templates -> Visual C# -> Web -> ASP.Net Web Application -> Web API
2. Cài đặt EntityFramework nếu như chưa có.
3. Tạo 2 Entity Model như sau:
   1. class Chat
      1. Id: int, PrimaryKey, Identity(1,1)
      2. Content: string
      3. Username: string, ForeignKey nối sang bảng User
      4. SentTime: DateTime
   2. class User
      1. Username: string, PrimaryKey
      2. Password: string
4. Tạo DataContext với DbSet để kết nối dữ liệu
5. Tạo DataInitializer với 1 vài dữ liệu mẫu để demo và test.
6. Tạo 2 controller tương ứng với model đã tạo:
   1. UsersController: Bao gồm các phương thức:
      1. bool Get(string Username, string Password): Sử dụng như chức năng Login, nếu có User có cùng Username và Password trong database thì trả về true, nếu không thì trả về false.
      2. String Register(string Username, string Password): Sử dụng như chức năng Register, thêm mới User với Username và Password được cung cấp vào cơ sở dữ liệu, kiểm tra tránh trùng Username và các xử lý lỗi khác. Trả về chuỗi thông báo phù hợp tương ứng.
   2. ChatsController: Bao gồm các phương thức:
      1. List<Chat> Get(): Trả về tất cả Chat có trong CSDL.
      2. List<Chat> Get(int id): Trả về tất cả Chat có Id>id được truyền vào.
      3. Void Post(string Content, string Username): Thêm mới Chat với Content và Username được cung cấp, SentTime đặt bằng thời gian hiện tại của máy chủ.
7. Tạo MVC Project, consume WebAPI như sau:
   1. Trên Home, tạo Action và View Login. Trên View tạo 2 textbox tương ứng là Username và Password, cùng 1 nút Submit Login. Khi người dùng nhập đúng thông tin đăng nhập và Login, thông qua HttpClient gọi đến UsersController, phương thức Get(Username, Password) để kiểm tra thông tin Login có chính xác không, nếu đúng thì tạo Session[“LoginUser”] = Username, sau đó cho chuyển sang Action / View Chat, nếu sai thì trả về thông báo lỗi lên giao diện Login.
   2. Trên Home, tạo Action và View Register, với giao diện tương tự Login. Khi người dùng Submit thông tin, thông qua HttpClient để gọi đến UsersController, phương thức Post(Username, Password) để thêm mới User. Nếu đăng ký thành công thì tạo Session[“LoginUser”] = Username và đẩy sang Action / View Chat. Nếu không thì hiện thông báo lên giao diện.
   3. Trên Home, tạo Action và View Chat, trên View cho TextArea Chat để hiển thị tất cả nội dung chat hiện có. Bên dưới là 1 textbox Message và nút Send để nhập và gửi nội dung Chat mới lên WebAPI. Khi chưa đăng nhập (sử dụng Session[“LoginUser”] cho phần đăng nhập, khi chưa đăng nhập thì session này bằng null hoặc empty), đẩy người dùng sang action Login trong HomeController. Khi đã đăng nhập, hiện View.
   4. Dùng Jquery + Ajax để lấy dữ liệu Chat từ ChatsController, phương thức Get. Sau khi lấy được dữ liệu, lặp từng phần tử và hiển thị vào thành 1 dòng trên TextArea Chat. Lưu lại Id lớn nhất của dữ liệu.
   5. Dùng Jquery + Ajax, cứ mỗi 1 giây lại truy vấn đến Web API, Chats Controller, phương thức Get(id) trong đó Id là biến đã được lưu từ bước d. Nếu có dữ liệu mới, chạy lặp từng phần tử và tiếp tục thêm vào TextArea Chat. Nếu không thì bỏ qua.
   6. Dùng Jquery + Ajax, mỗi khi người dùng bấm nút Send lại gửi lên WebAPI nội dung Chat từ textbox Message và Username của người dùng đã đăng nhập (Lấy qua Session vào biến Javascript).
8. Kết luận: Web API có những lợi thế khác so với WCF, trong trường hợp này có thể dễ dàng kết hợp hơn với MVC và Ajax để lấy dữ liệu liên tục và chỉ lấy dữ liệu mới, tránh gây tổn hao băng thông qua đường truyền. Tuy nhiên, cũng nhấn mạnh WCF cũng có thể làm được việc trên nhưng sẽ phức tạp hơn so với WebAPI.
9. Có thể giới thiệu thêm 1 số cách khác để tạo ứng dụng Chat hay các ứng dụng online thời gian thực khác trên nền web mà không cần sử dụng Web API như sử dụng SignalR.