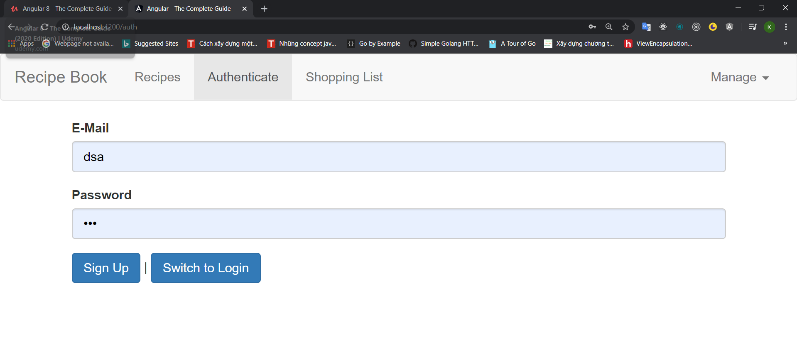
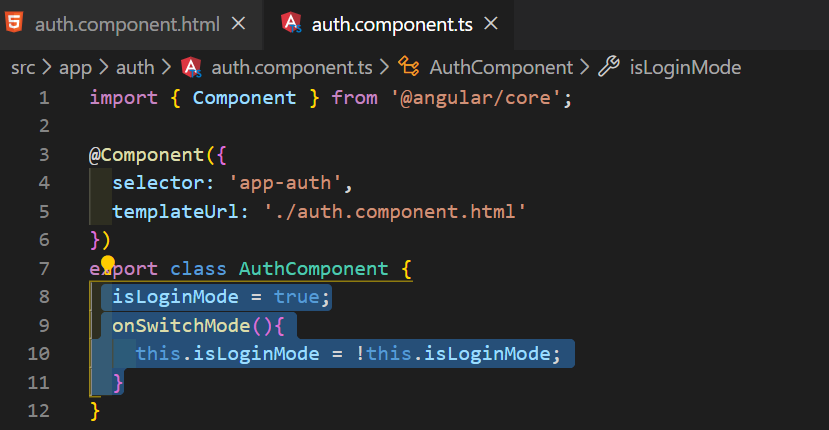
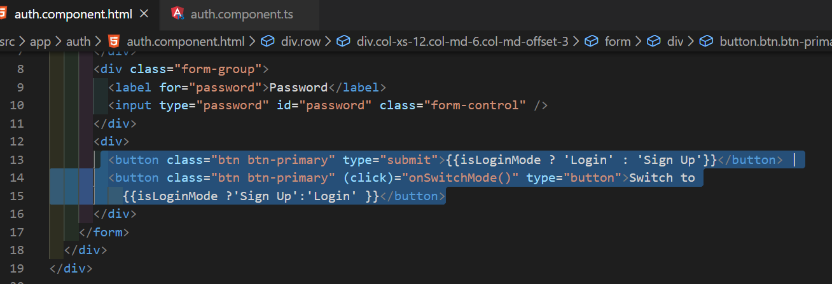
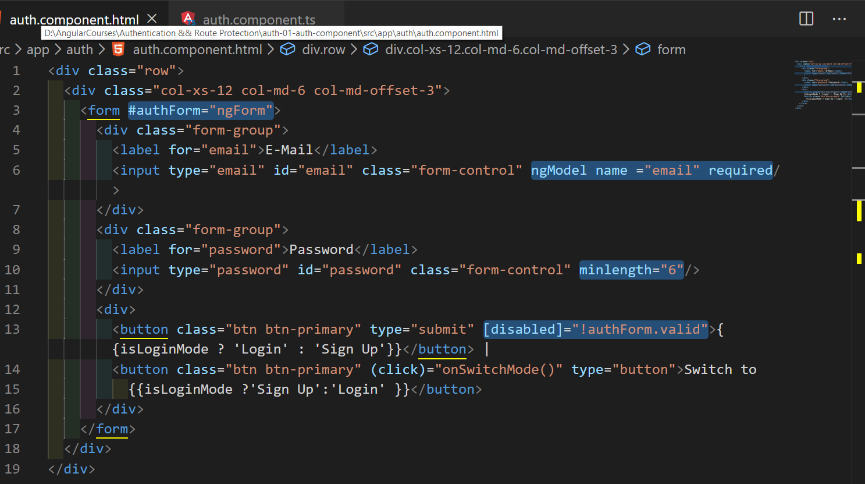
1. **SECTION 20: Authentication & Route Protection in Angular (** là mấy cái log-in, log-out, user, admin, xác thực người dùng….): **Tạo 1 folder Auth để quản lý vấn đề login, logout, xác thực user…**
   1. **Switch between Auth Modes:** Nhấp vào nút ‘Sign Up’ thì nút ‘Sign Up’ biến thành ‘Log In’, chuyển state isLoginMode = !isLoginMode

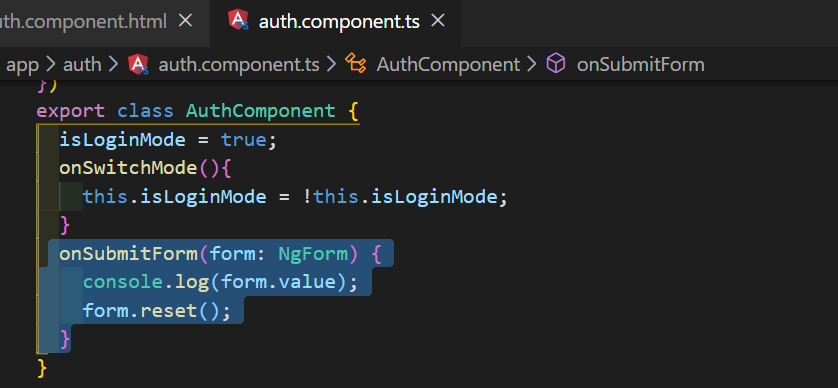
**** Giao diện Authentication.



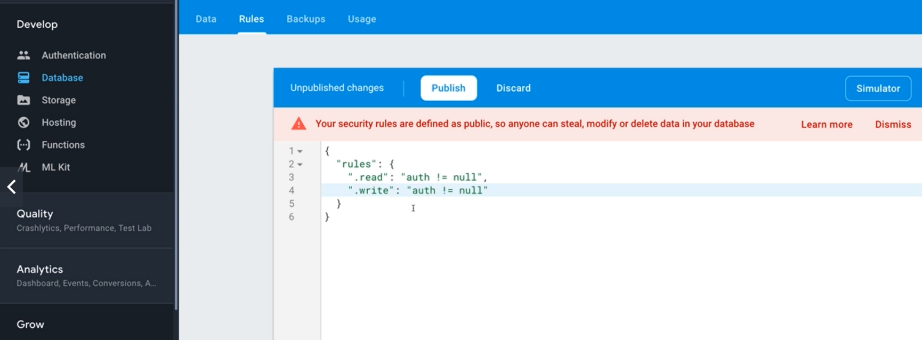


* 1. **Handling Form Input: (**Làm mấy cái đã học trong Section Form**):**

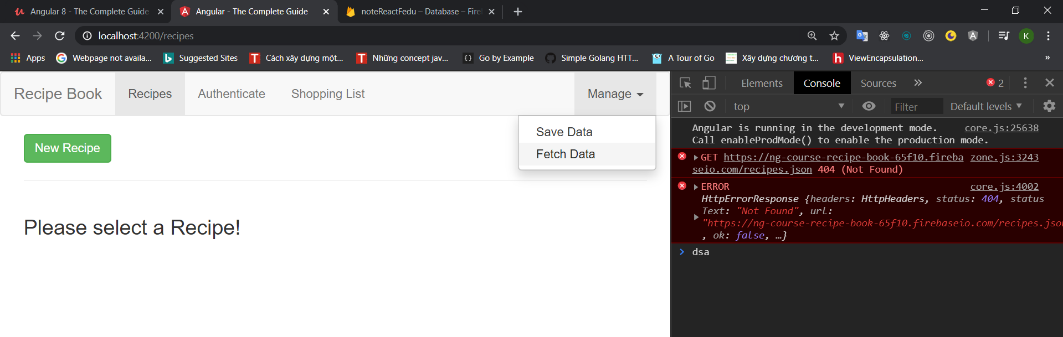
****

****Nhấp vào nút Submit thì lấy được value của ‘email’ và ‘password’ để chuẩn bị gửi về server thể xác thực.

* 1. **Preparing the Back-end:** mình sẽ làm việc vs firebase vì nó đơn giản.

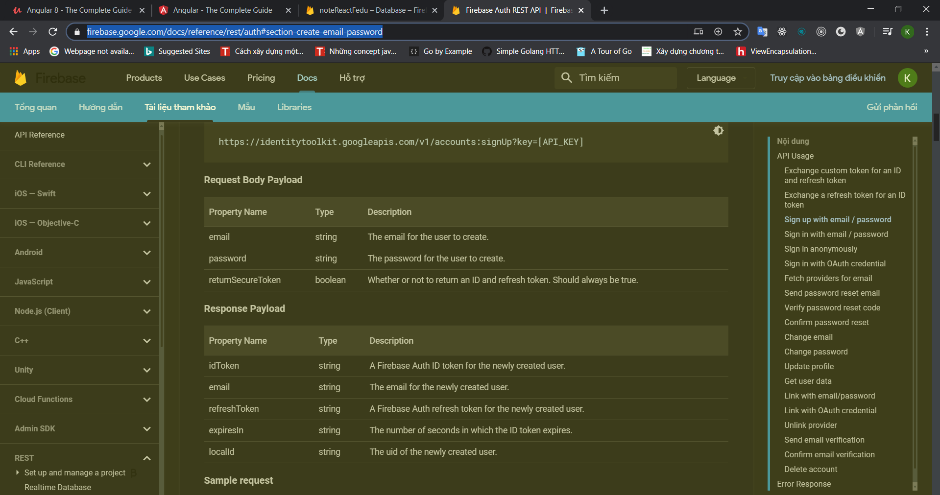


-Chỉnh từ ‘True’ sang ‘Auth != null”, nghĩa là chỉ những người dùng đã được xác thực (authenticated Users) mới được đọc và ghi dữ liệu.

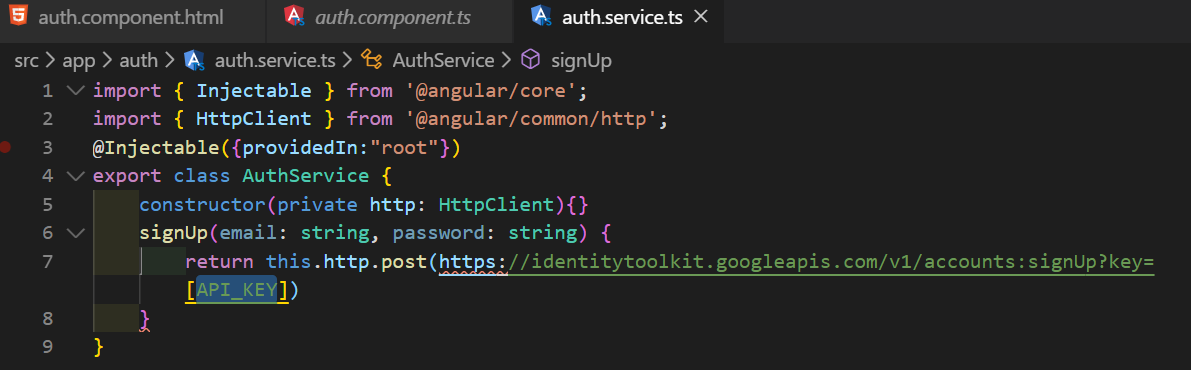


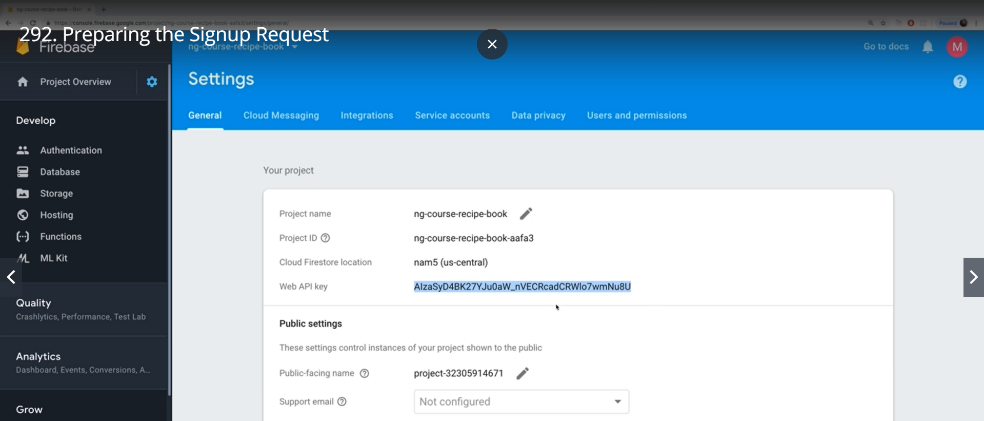
-Lúc này khi ‘Fetch Data’ thì bị lỗi 404( là không tìm thấy được)

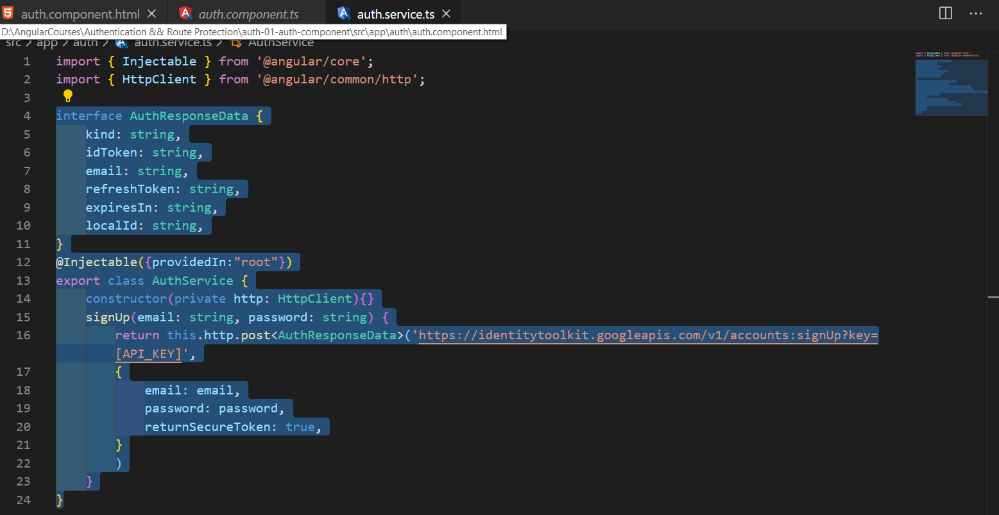
* 1. **Preparing the SignUp Request:**



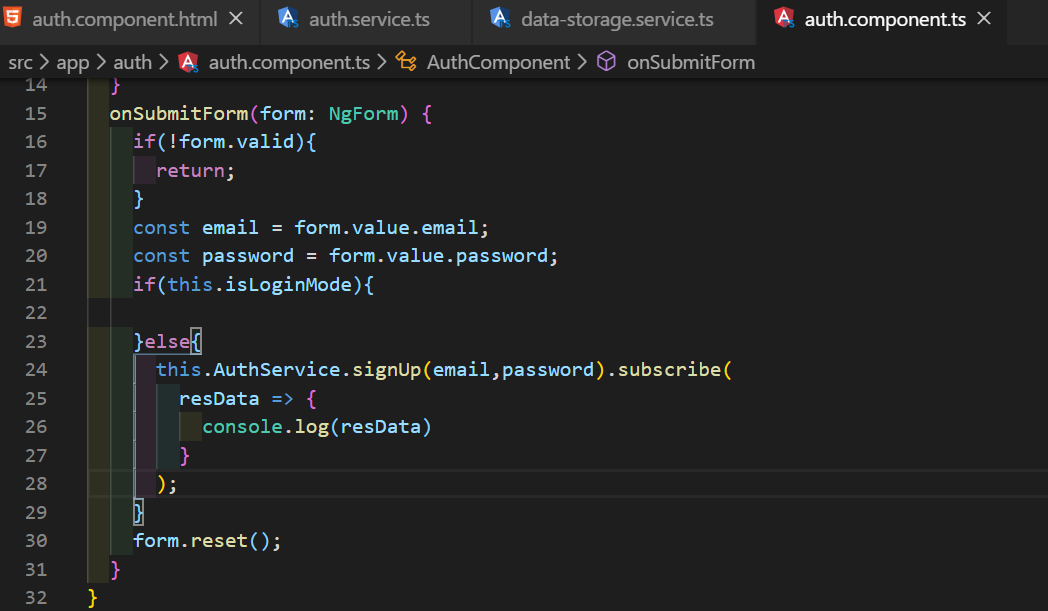
* <https://firebase.google.com/docs/reference/rest/auth#section-create-email-password>

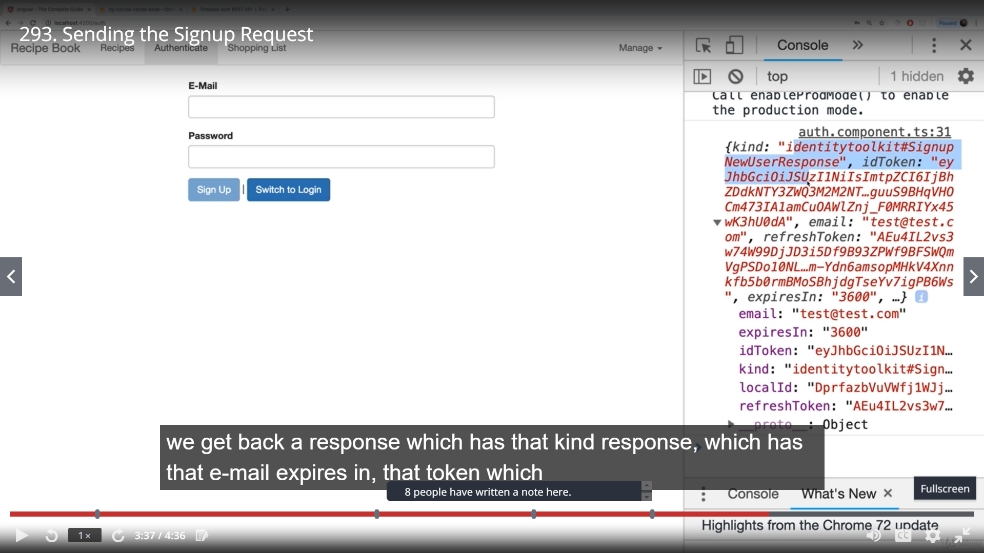
-Api Key lấy trong Firebase

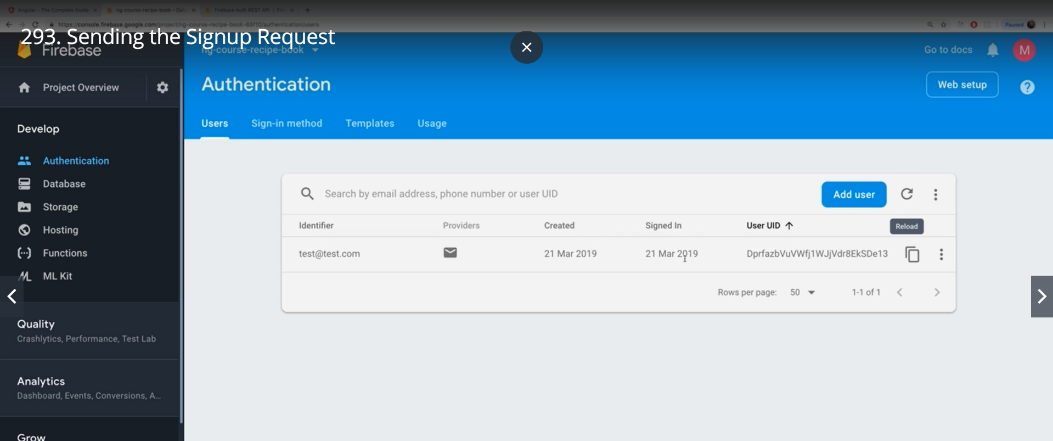


-Đường link kia là API để gọi.Còn kiểu trả về hay kiểu Request email password là dựa theo yêu cầu của API.

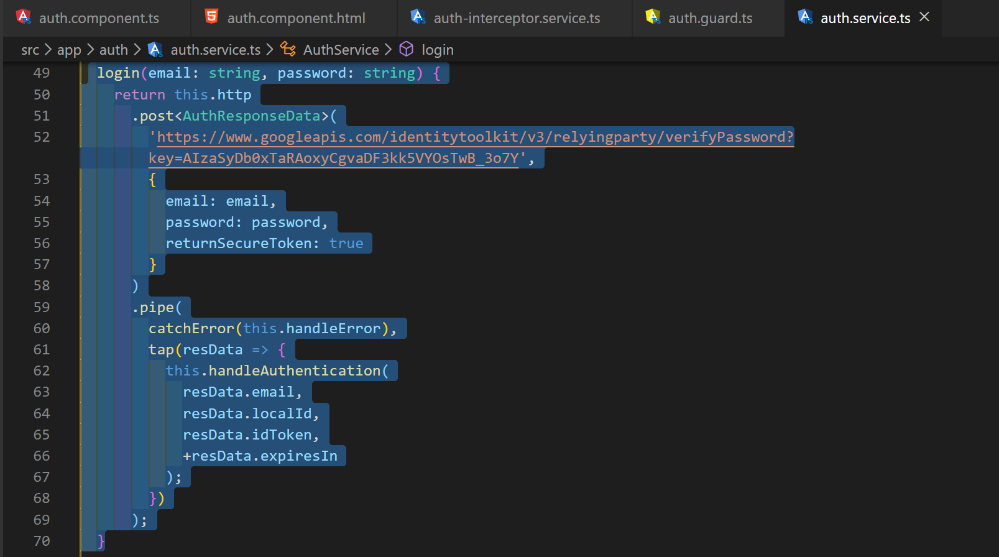
* 1. **Sending the SignUp Request**

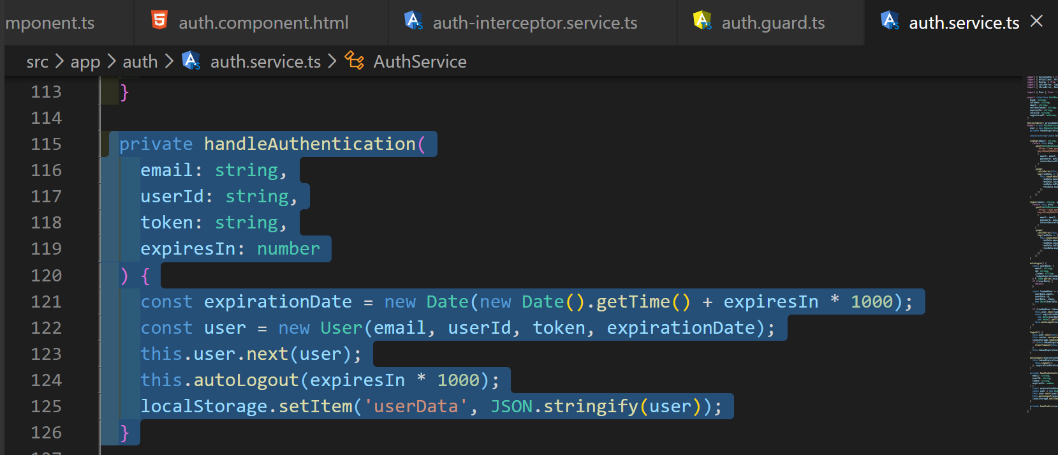


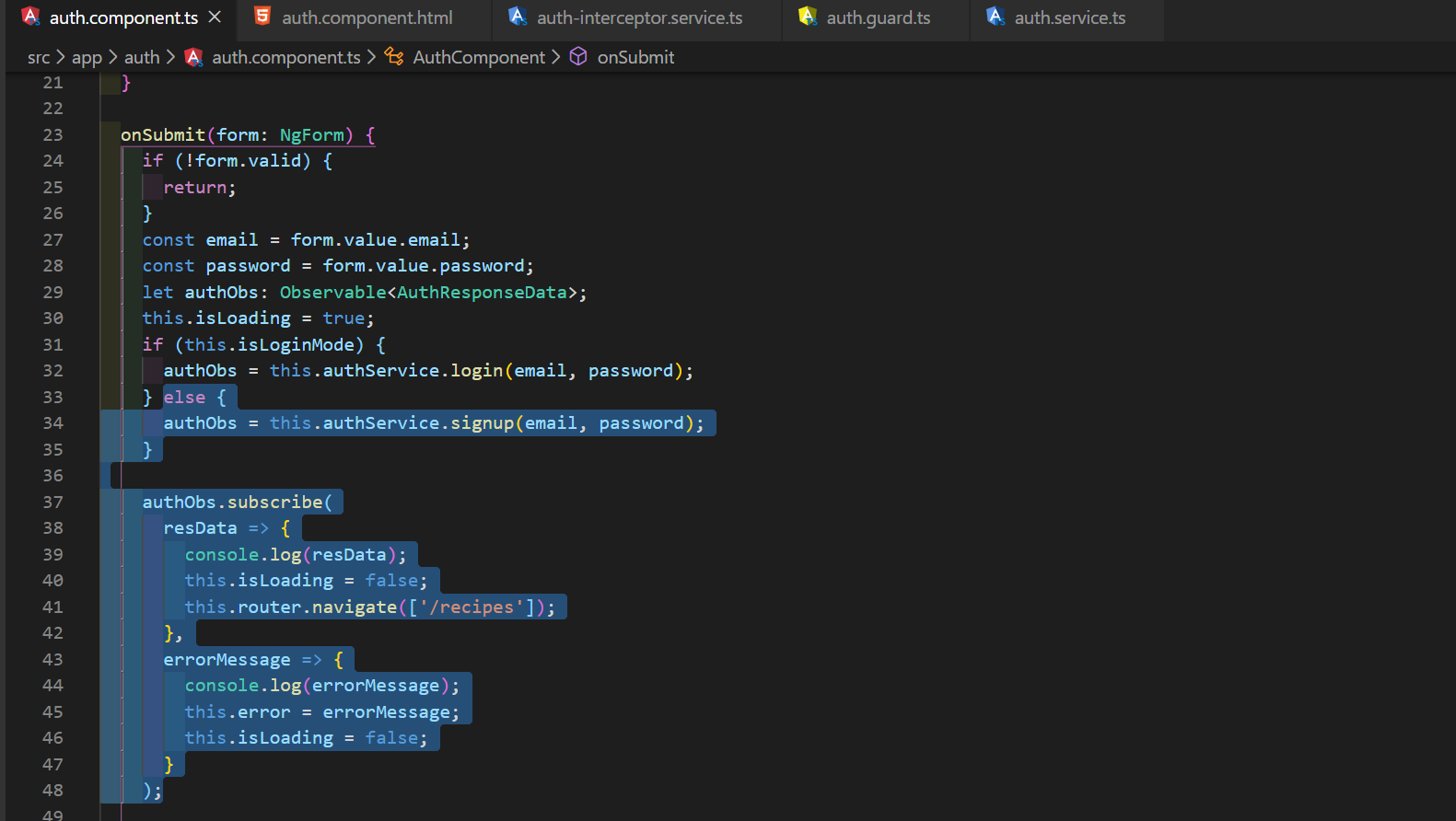
Đây là ResponseData được trả về.

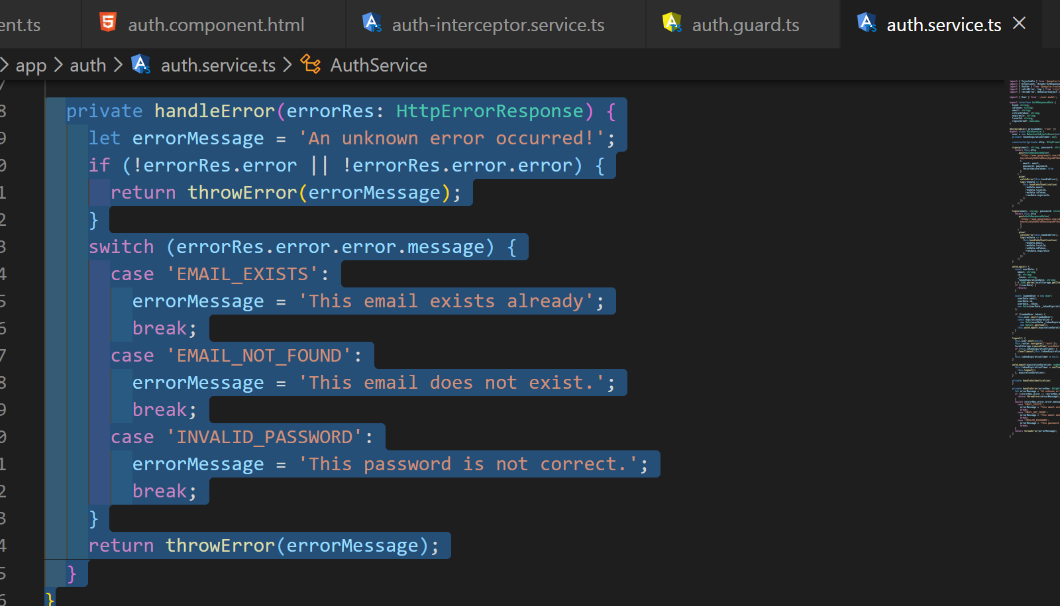
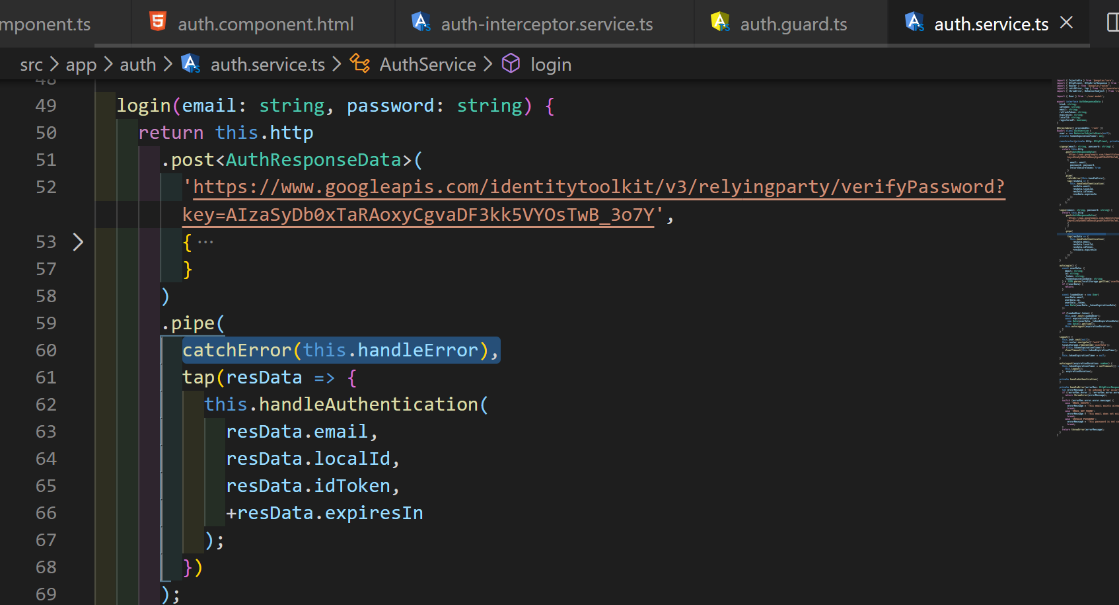
-Database được cập nhật khi SignUp(đăng ký) thành công.

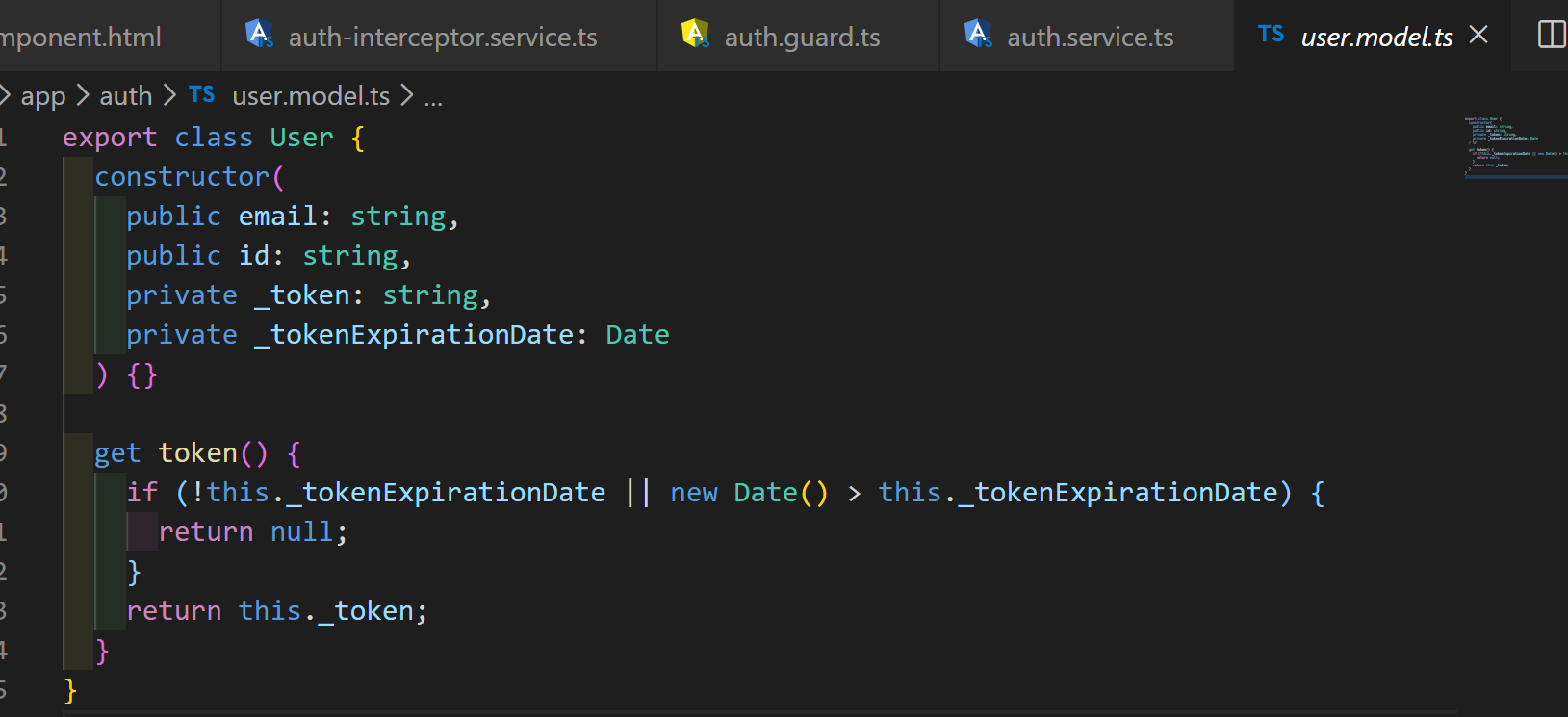
* 1. **Sending Login Requests**



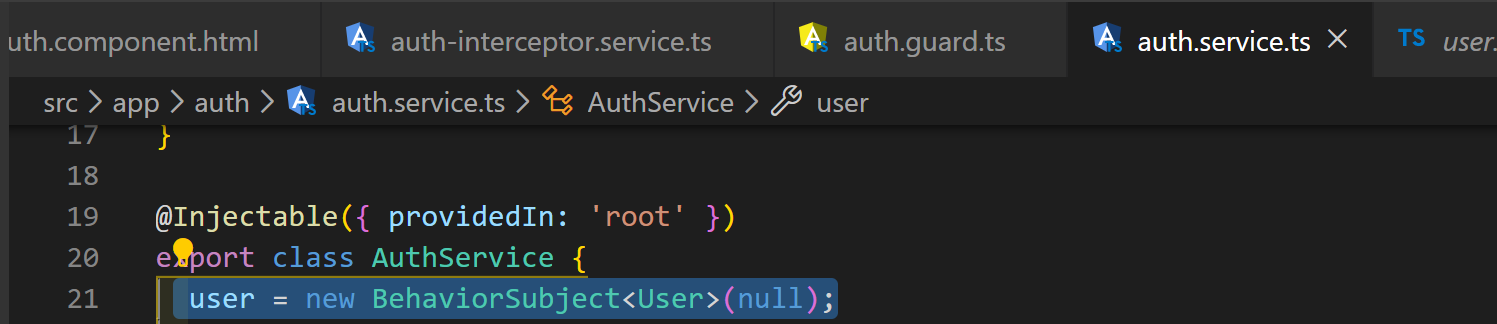


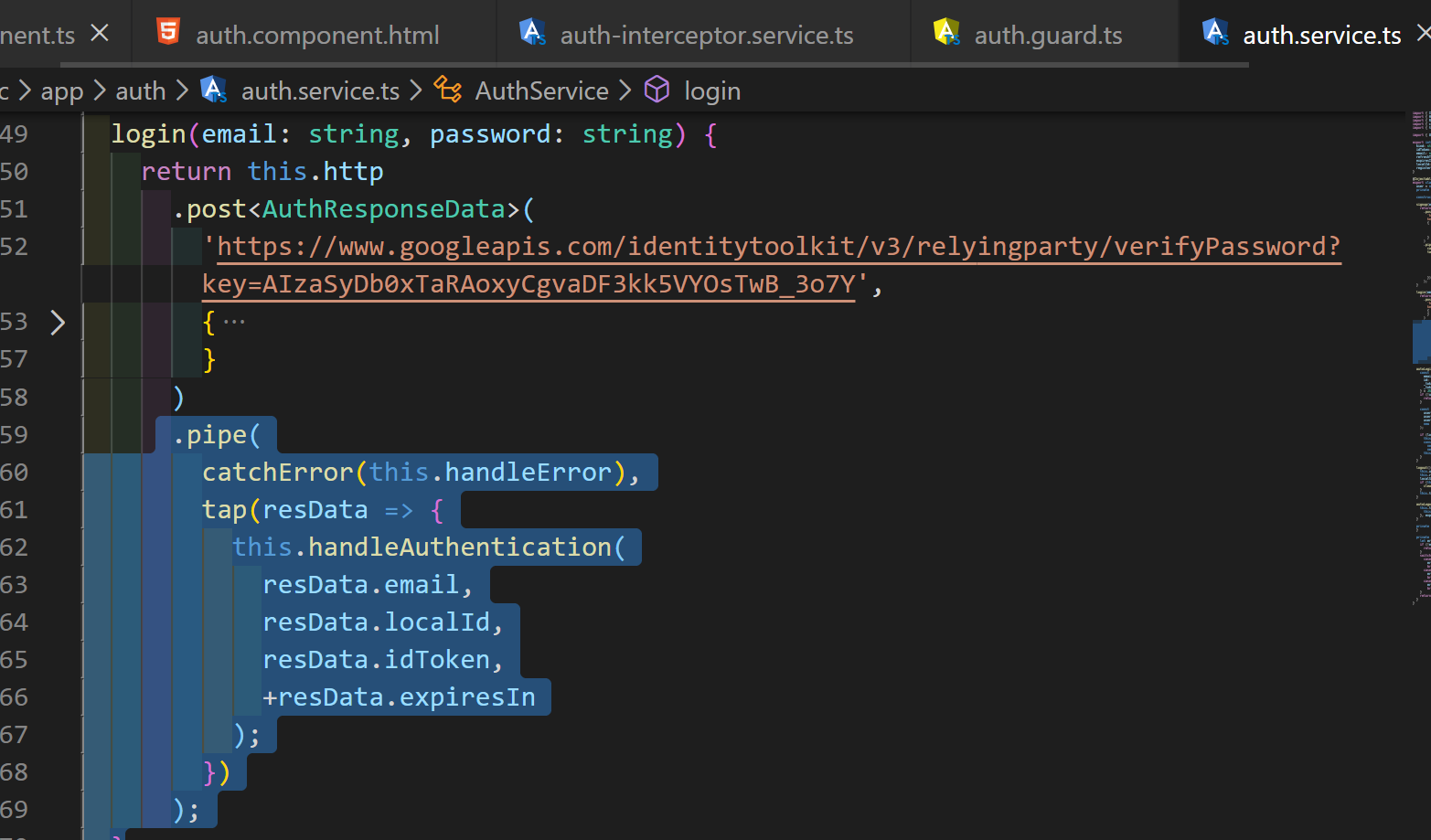
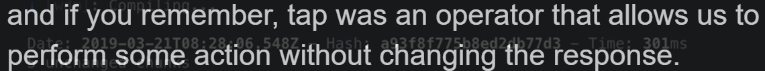


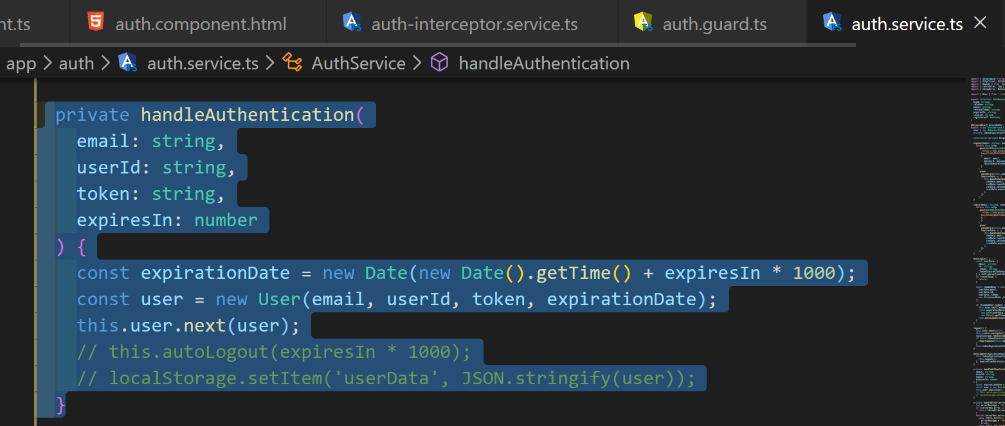
* 1. **Login Error Handling:** giải quyết khi có quá trình login xảy ra lỗi, như tên đăng nhập, password sai…
     1.  -Khai báo hàm để xử lý khi Error xuất hiện trong quá trình login, sign up.
     2.  -Khi catch được Error sẽ gọi hàm handleError() Trên để giải quyết, để xuất ra Output.
  2. **Creating and Storing the user Data:** Lưu trữ thông tin của user, xác định xem user có được xác thực không, …
     1. **Tạo 1 file user-model.ts**: dùng để lưu trữ tất cả thông tin của 1 user và xác định xem token của user đó có hợp lệ hay không

 -Các thông tin của user tùy thuộc vào API trả về như thế nào. Tùy vào Back-end thiết lập.

-Hàm get token() , nếu không có mấy cái chức năng tự xóa token sau 1h thì chỉ cần dòng return token là được.Không cần như trên.

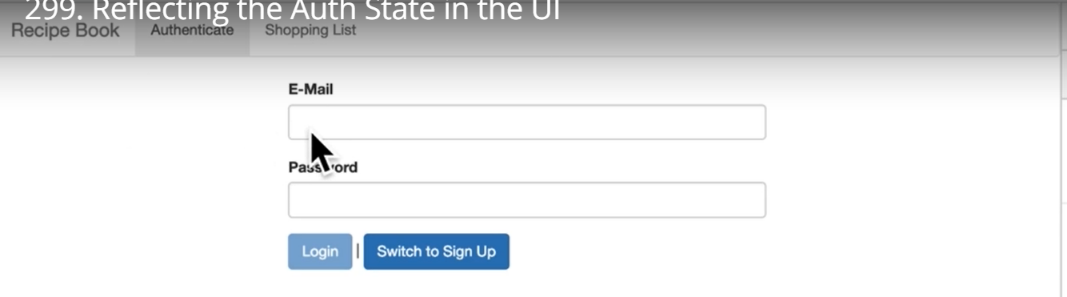
1.8.2. - Chúng ta khai báo 1 User ở service để quản lý user. Khi nào ta muốn emit 1 new user thì ta next() a new user, bất cứ khi nào ta log-in, log-out, ta clear the user ….

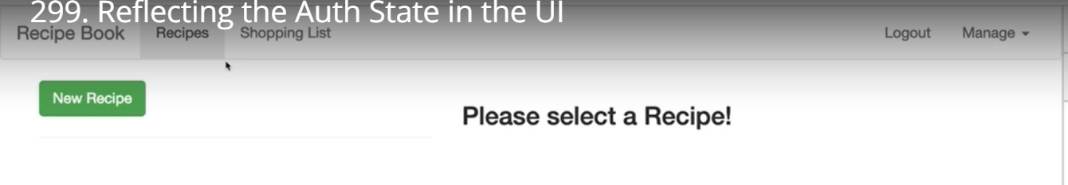
1.8.3.  -Dùng operators Tap() để log in User, hàm Tap() .

 -Dùng để emit 1 user mới khi có ai đó sign-up hoặc log-in.Nếu không làm chức năng tự động log-out thì không cần làm dòng const ExpirationDate=….

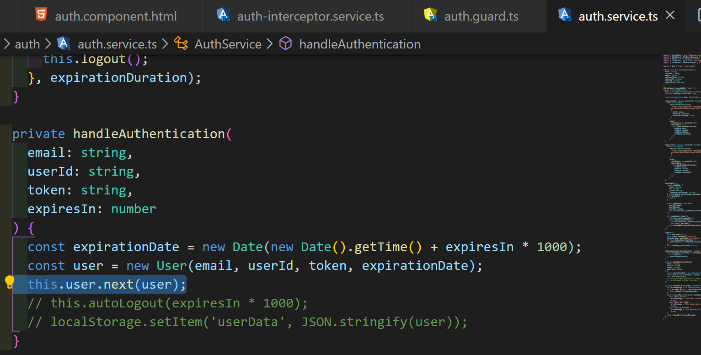
**GIẢI THÍCH**: Khi ta log-in hoặc sign-up, ta gửi 1 request đi có pass, username, ta sẽ nhận về 1 response gồm các thông tin như email, userId,token(tùy vào back-end do ta thiết kế).Sau khi nhận được response, ta dùng hàm tap() để làm 1 số thao tác dựa trên response mà ta vừa lấy như tạo mới,emit,next() và lưu trữ user như hàm handleAuthentication() đã làm như trên.

* 1. **Reflecting the Auth State in UI:** Nghĩa là ta sẽ dùng các authenticated state để chỉnh sửa, thiết lập UI cho hợp lý. (ví dụ như khi vừa log-in thì hiện lên chữ log-out,khi log-in rồi thì mất logInComponent..)
     1. **Ví dụ**: đây là 2 header của page khi chưa log-in và đã log-in thành công.

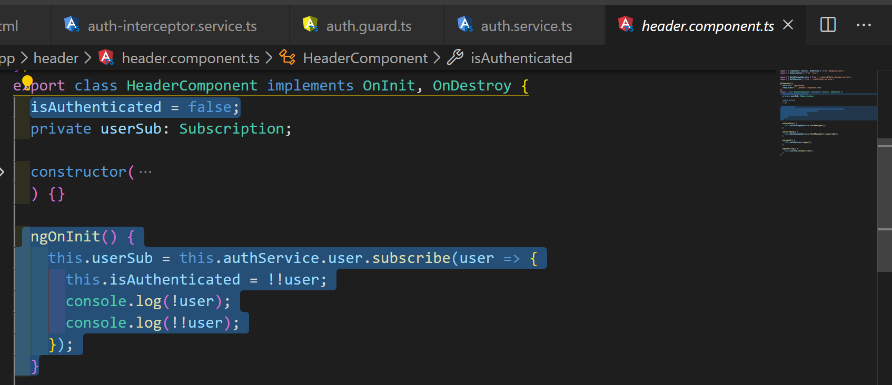
**** -HeaderComponent khi chưa log-in

**** **-**HeaderComponent khi đã log-in thành công.

* + 1. **Cách làm** ví dụ trên: làm Tương tự với các component khác. (thay đổi giao diện UI dựa vào User đã được log-in hay chưa.):

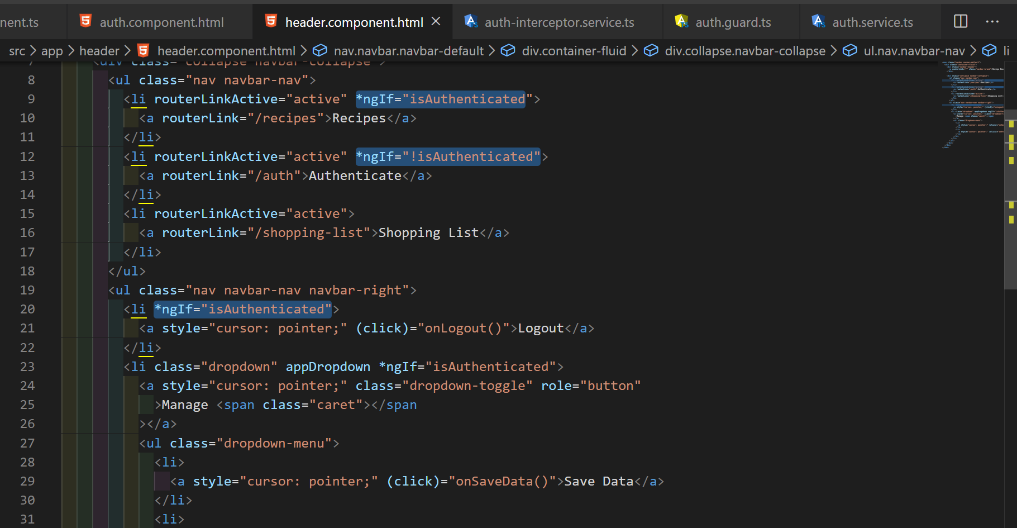
-Hãy nhớ khi ta submit log-in hoặc sign-up, ta kích hoạt, emit, khởi tạo 1 user mới bằng hàm next().

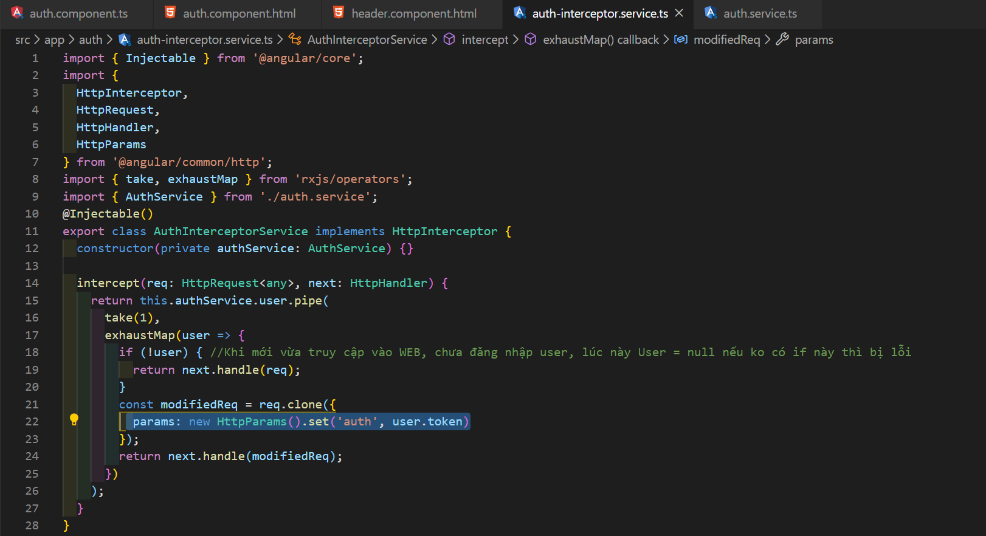
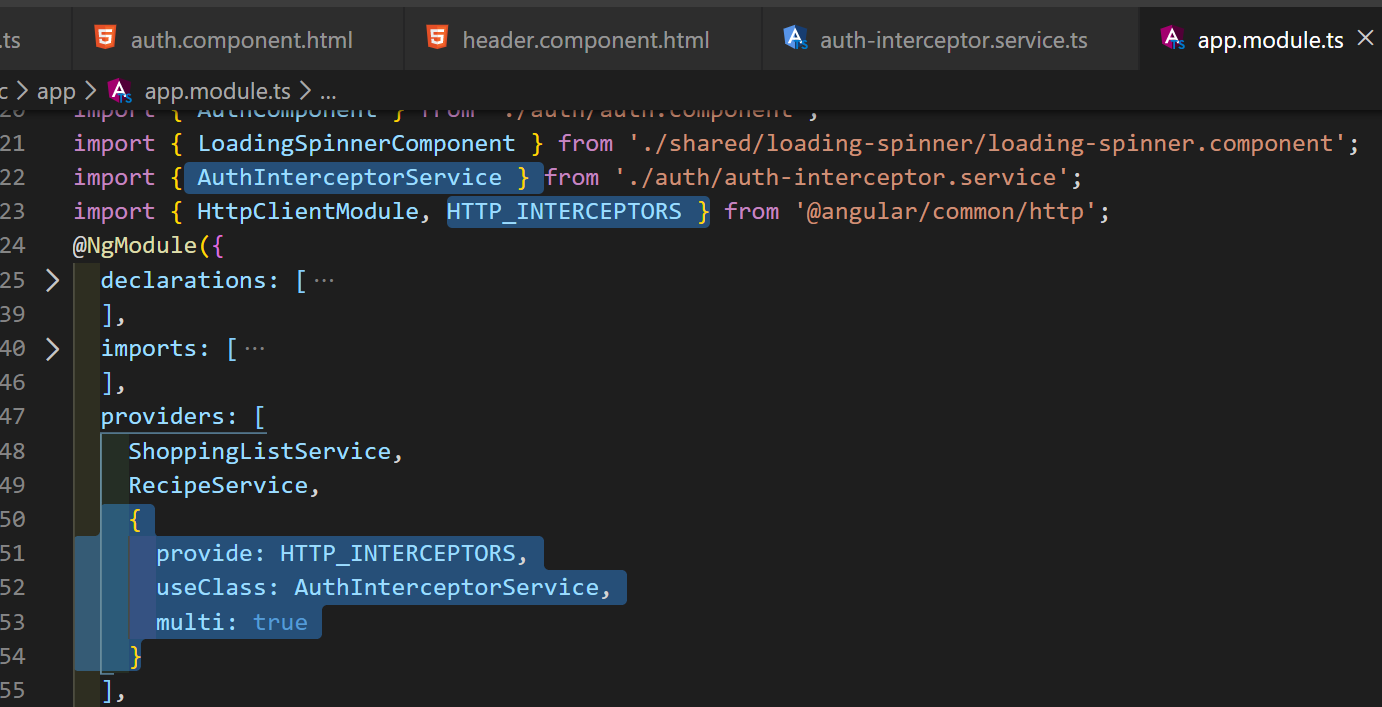
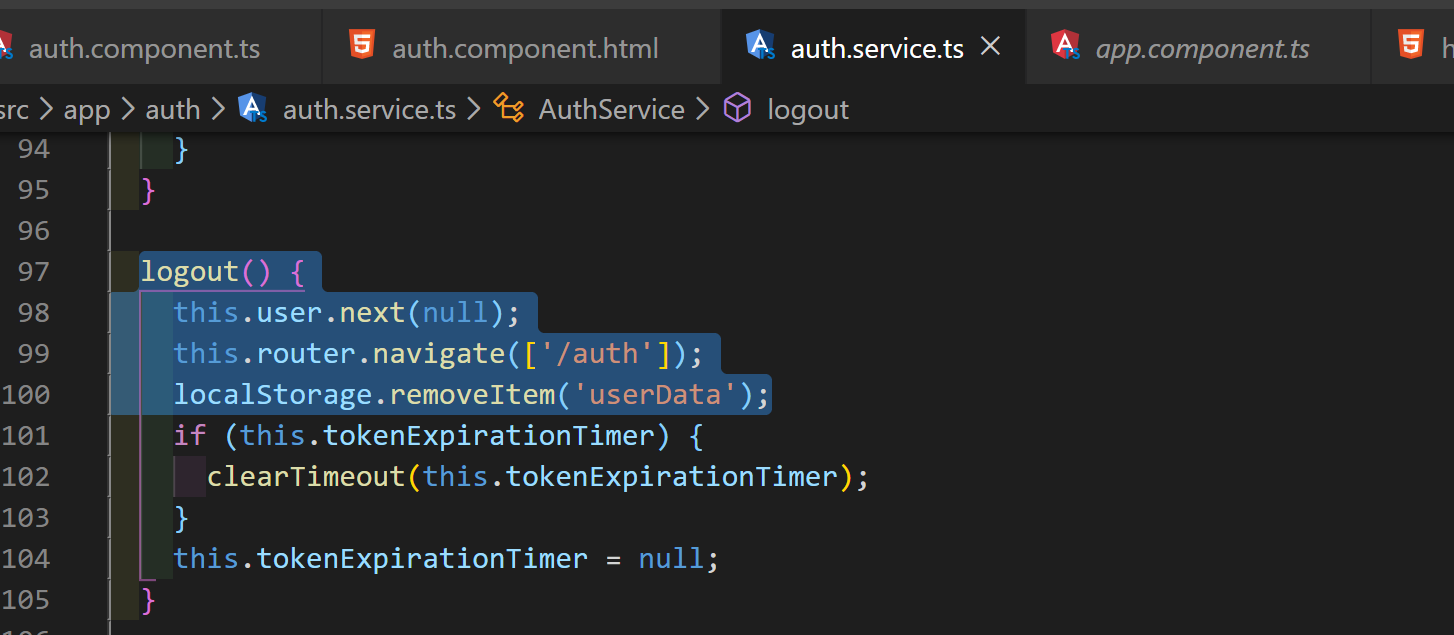
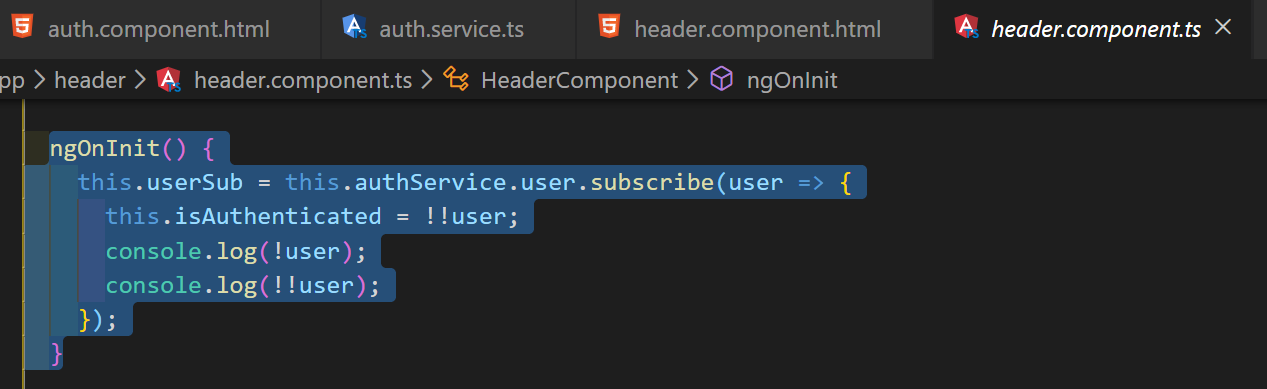
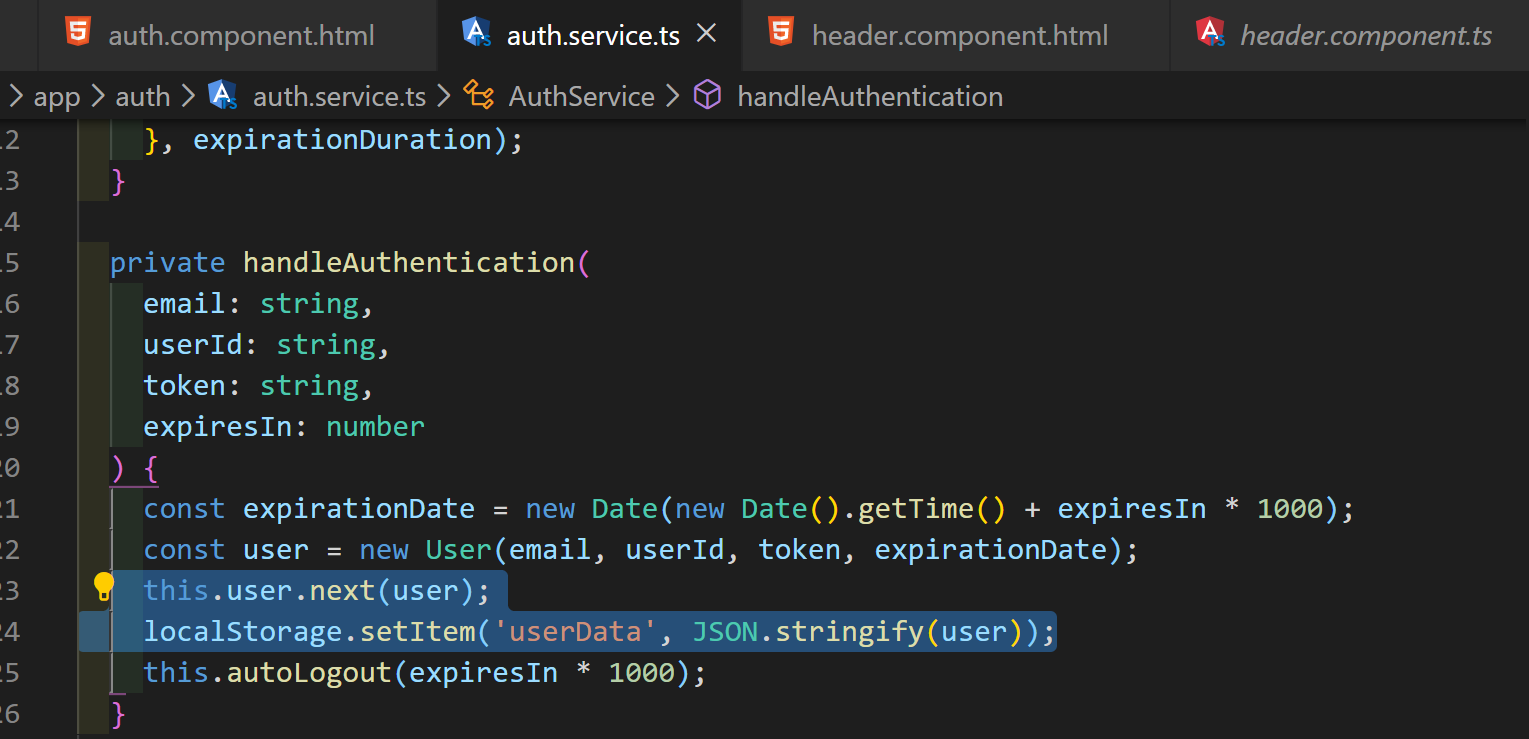
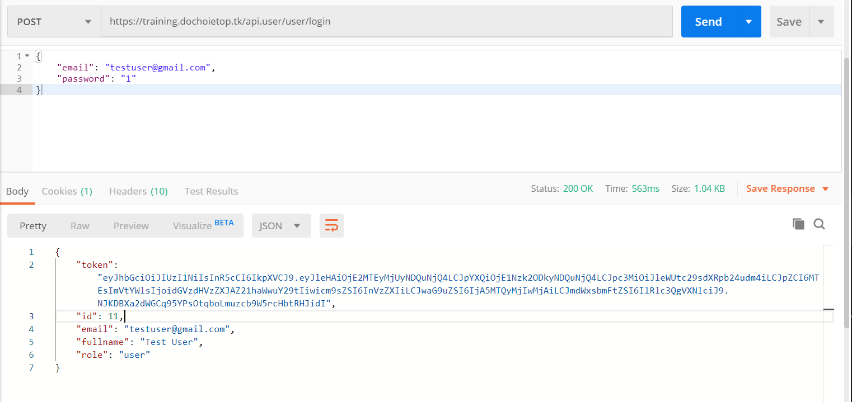
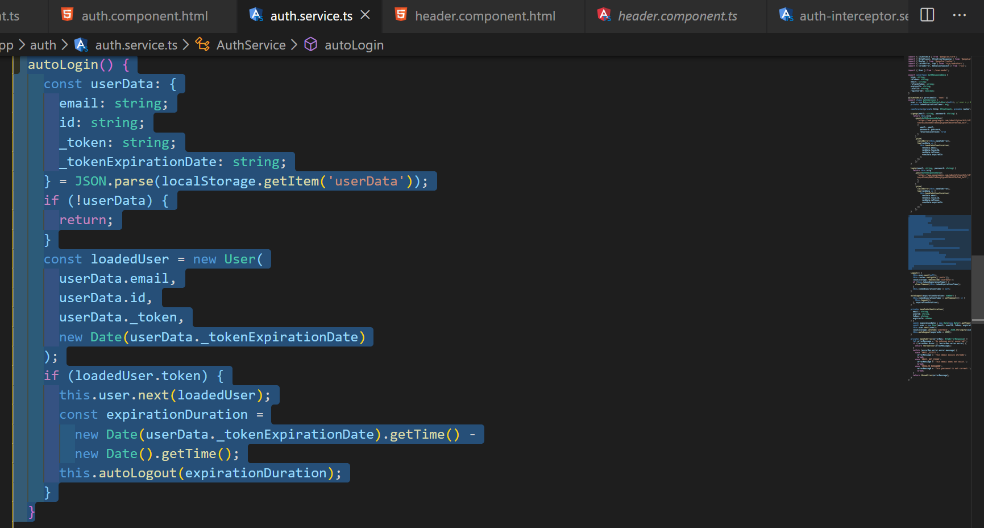
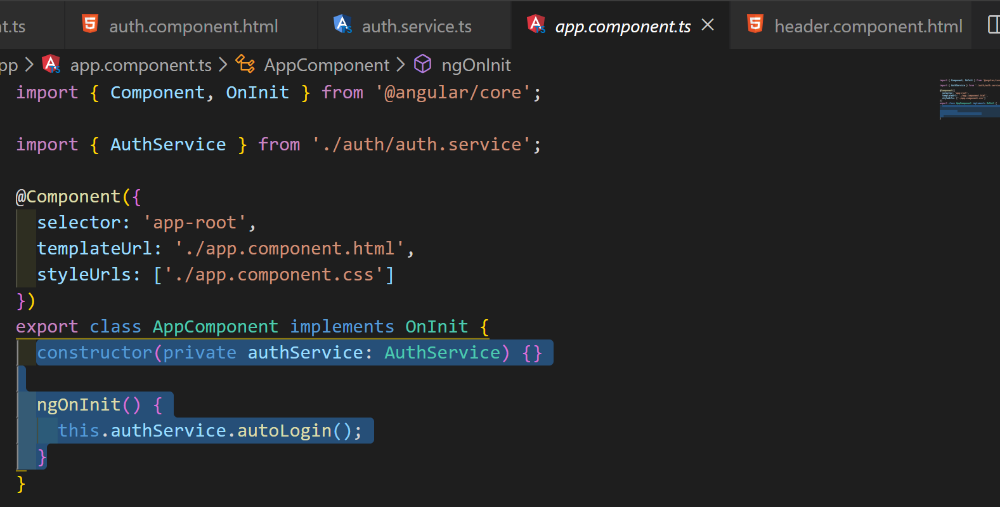
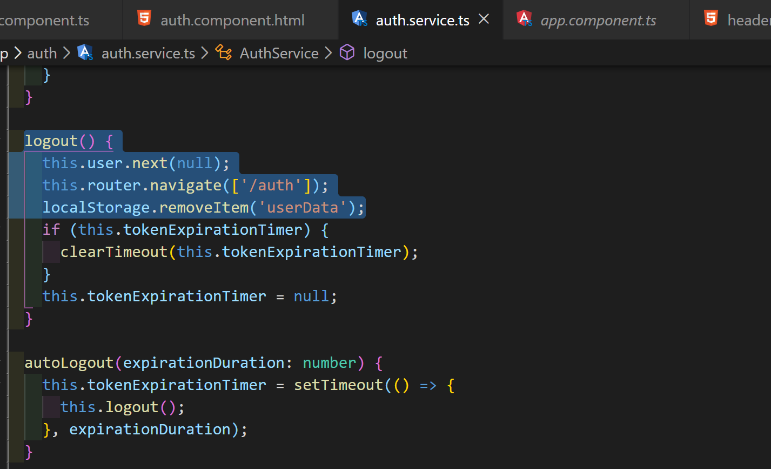
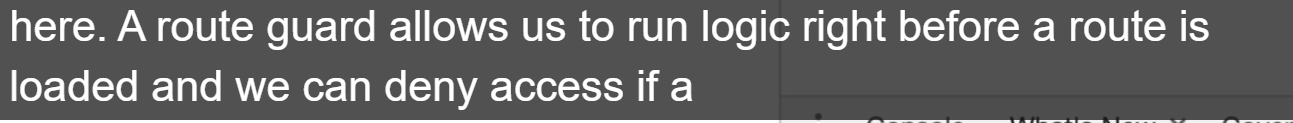
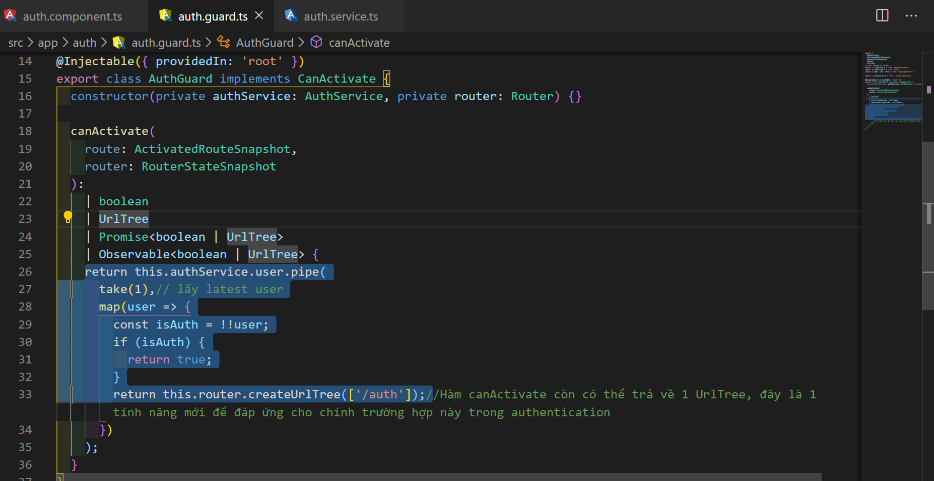
- nghĩa là mỗi khi submit thì this.user sẽ **bị thay đổi**, được kích hoạt lại, mà đây là 1 subject, nên ta có thể dùng hàm subcribe() để theo dõi sự thay đổi của nó trên toàn application.

****-Theo như ví dụ, ta thay đổi giao diện UI dựa theo sự thay đổi của this.user ở hình trên mỗi khi được emit.

**-**Nghĩa là mỗi khi có 1 user mới được emit(), subject User bị thay đổi, mà hàm subcribe() này là dùng để theo dõi sự thay đổi đó, mỗi lần nó thay đổi, nó sẽ cập nhật lại property isAuthenticated 1 cách chính xác để hiển thị ra UI 1 cách chính xác.

**-** nhớ onDestroy() nó.

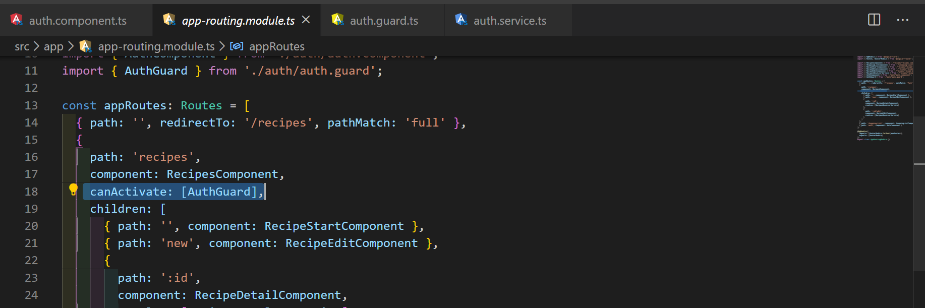
-Giao diện của UI thay đổi dựa trên sự thay đổi của user thông qua property isAuthenticated.

* 1. **Adding the Token to Outgoing Requests:** Khi đăng nhập vào, chúng ta cần phải lấy (fetch) data về từ Back-end, ta cần phải gửi token kèm theo Fetch Data Requests để BE biết là đây là 1 token hợp lệ.(**Sử dụng INTERCEPTORS**)**:**
     1. **Tạo file auth-interceptors.service.ts: -**Nghĩa là trước mỗi khi gửi 1 request đi, ta sẽ gửi kèm theo token của User ấy, nếu không có user nào log-in, không có token thì đó là trường hợp line 18.. Intercept() là chạy trước mõi khi 1 request được gửi đi trong toán app.
     2. **Thiết lập Interceptors:**
  2. **Adding Logout:** thêm chức năng Logout()
     1. -Khi click vào button Log-out của Header thì gọi hàm Log-out().
     2.  -viết hàm Logout() để Emit User về null khi ta click vào button Log-out.Thay đổi xong nó di chuyển về trang Đăng nhập.Nhớ xóa dataUser trong localStorage (Đó là code của chức năng auto log-in)
     3.  -Mỗi khi subject User thay đổi thì các hàm Subcribe() như trên cũng cập nhật lại các property để hiển thị Output cho chính xác.
  3. **Adding Auto-Login:** Nghĩa là vẫn giữ tình trạng log-in mỗi khi ta reload lại trang (dùng local Storage).
     1.  -Mỗi khi ta emit 1 user, thay đổi 1 user khi log-in hoặc sign-up, ta sẽ lưu userData ấy vào localStorage.(gồm có cả email, token, userName …. Nhiều lắm). UserData này là response được trả về chứ không phải là userData được gửi đi.-Đây là userData Response khi BE gửi về khi gửi Login API Request.
     2.  -Viết hàm này, để mỗi khi vừa reload lại trang, nó sẽ tự check xem trong LocalStorage có item userData không, Không có thì thui, **Có userData** thì tạo mới 1 user mới (bằng toán tử new), nếu User đó có **Token** hợp lệ thì emit, kích hoạt, thay đổi subject User ở Authentication. Nếu không cần các chức năng tự động Logout sau 1h thì khỏi cần mấy dòng dưới.
     3.  -Nghĩa là mỗi khi reload lại trang, là ta render lại <AppComponent> (lần đầu tiên khởi tạo Component), ta đều gọi hàm autologin() này, để check localStorage, check Token rồi tự động đăng nhập.
  4. **Adding auto-Logout:** Tự động thoát sau 1 khoảng thời gian.Tự xem lại 304.
     1. Coi thêm lại Src code.
  5. **Adding the Auth Guard: dùng** để cấp quyền truy cập vào cách route dựa trên các điều kiện do mình thiết lập.(Ví dụ như khi ta chưa Log-in thì ta không để truy cập vào Route [‘/abc’]).
     1. 

-Tạo file auth.guard.ts.

-hàm canActivate() cũng có thể trả về 1 URL tree.

-Nghĩa là trước khi ta truy cập vào component nào, guard đó kiểm tra xem có được phép vào không(line 30 là điều kiện kiểm tra), nếu được phép vào thì return true, nếu **không được phép vào** thì return false, hoặc có thể return về UrlTree để ta navigate qua route đó như line 33.

1.14.2. -Thiết lập, cài đặt AuthGuard

-Nghĩa là khi truy cập vào route =[‘/recipes’] thì ta sẽ được guard bảo về, thỏa điều kiện this.authService.user **tồn tại** thì được vào (return true), không thì return false hoặc urlTree(nagivate qua Routes khác.)

KẾT THÚC.