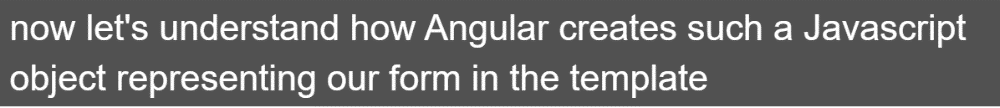
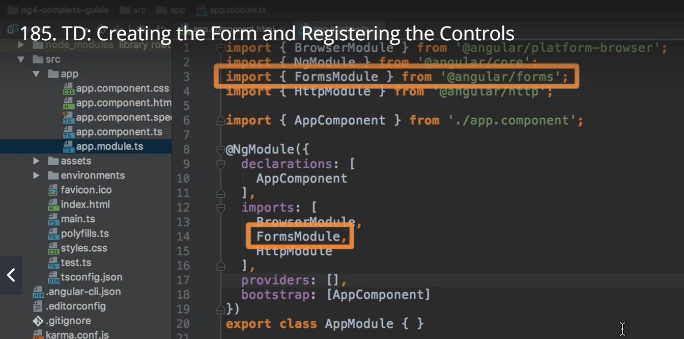
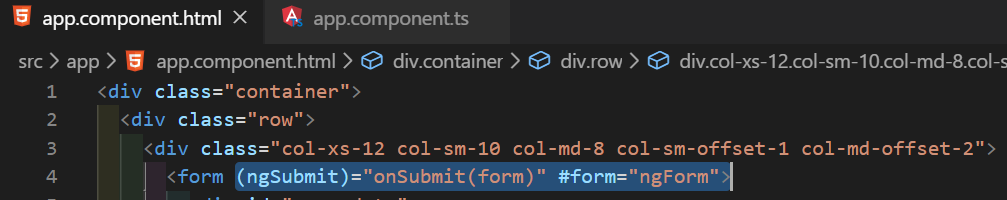
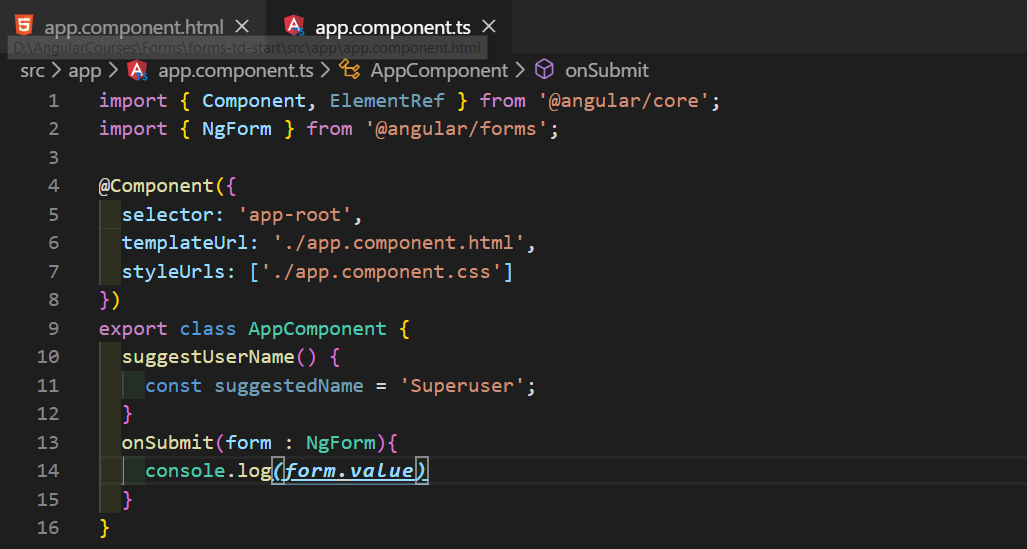
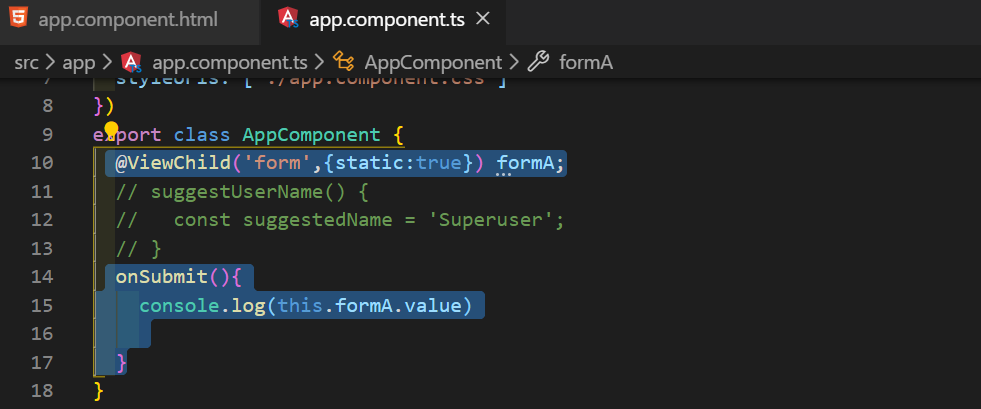
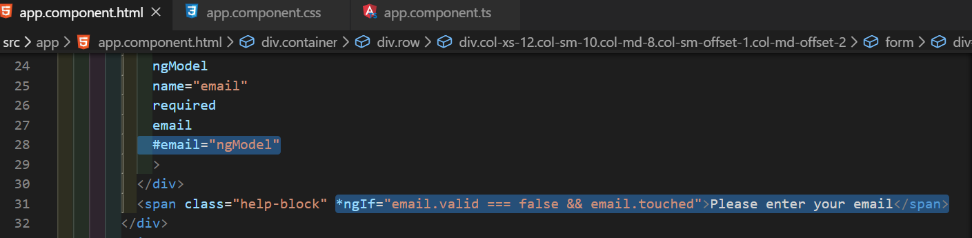
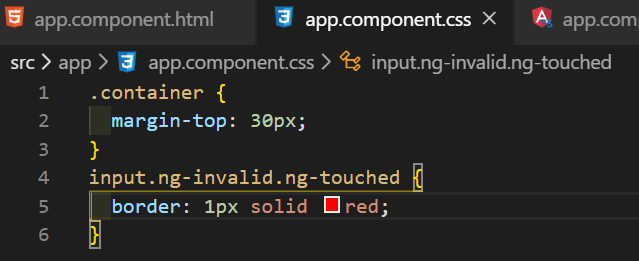
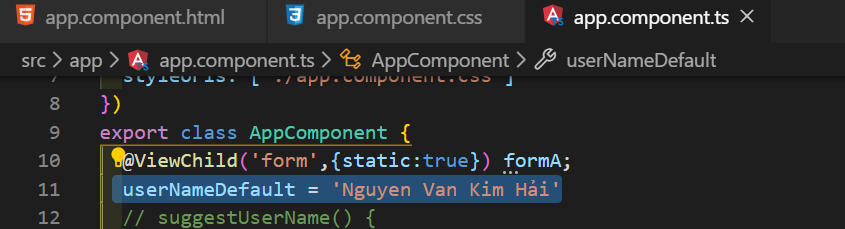
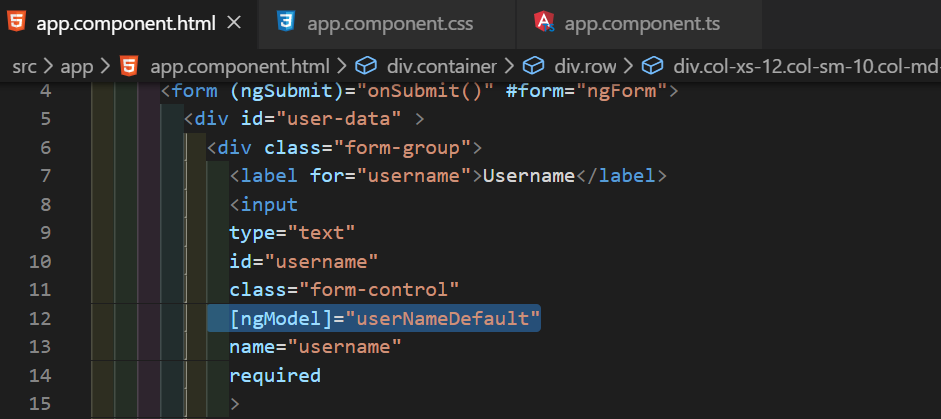
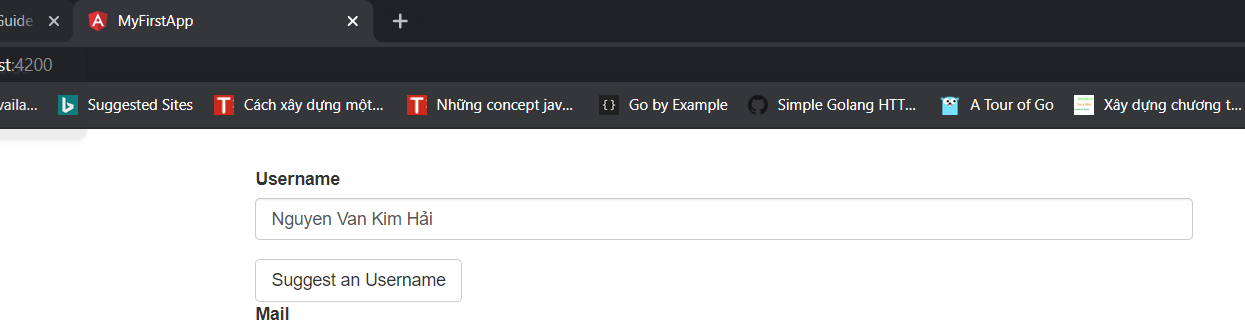
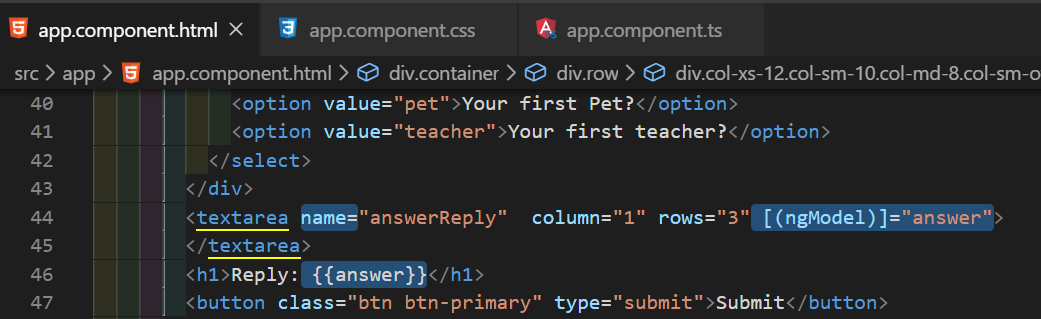
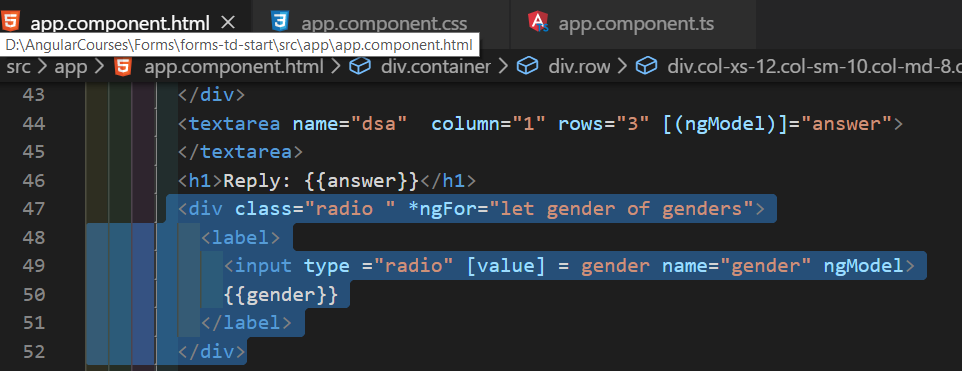
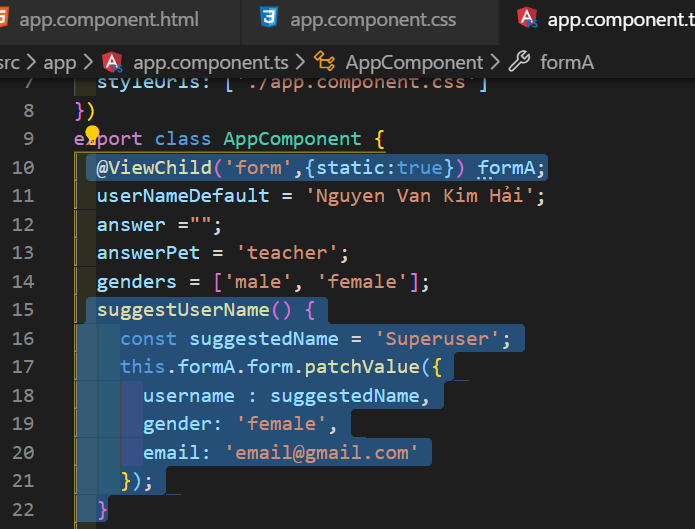
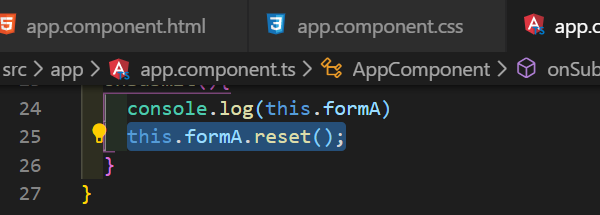
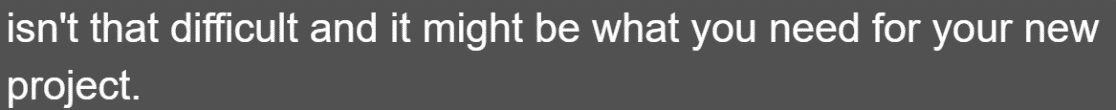
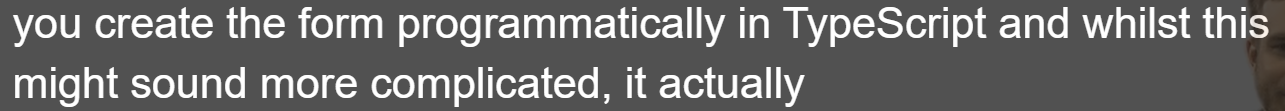
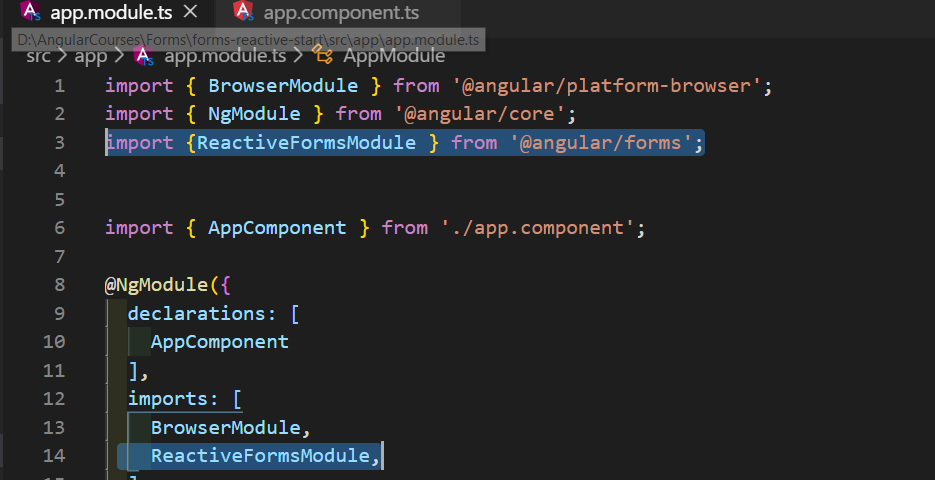
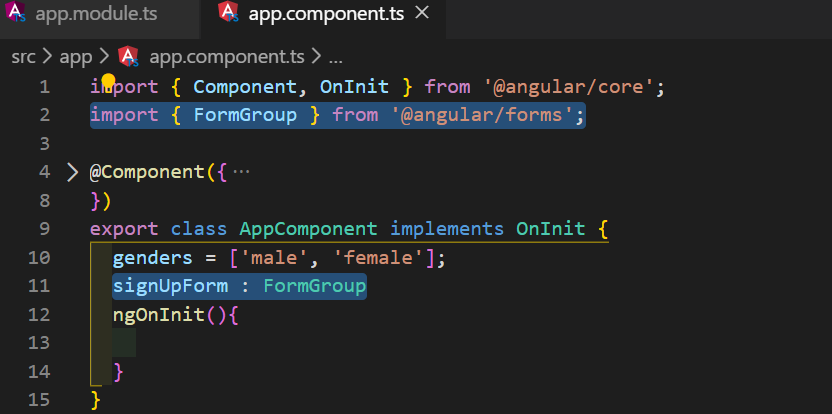
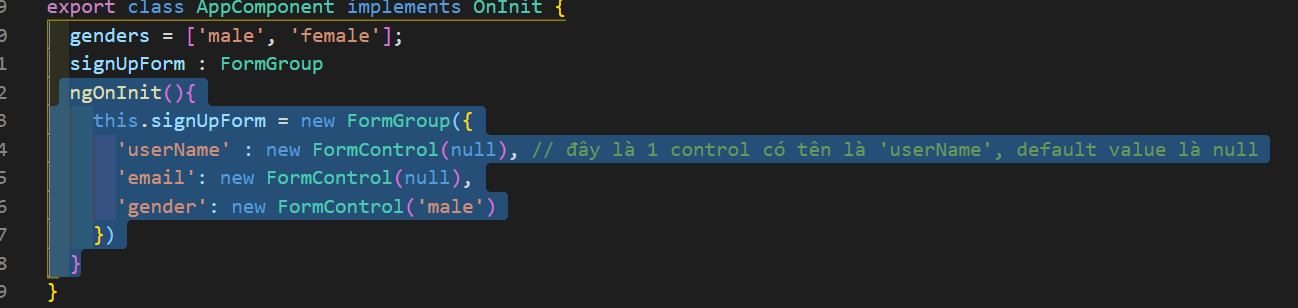
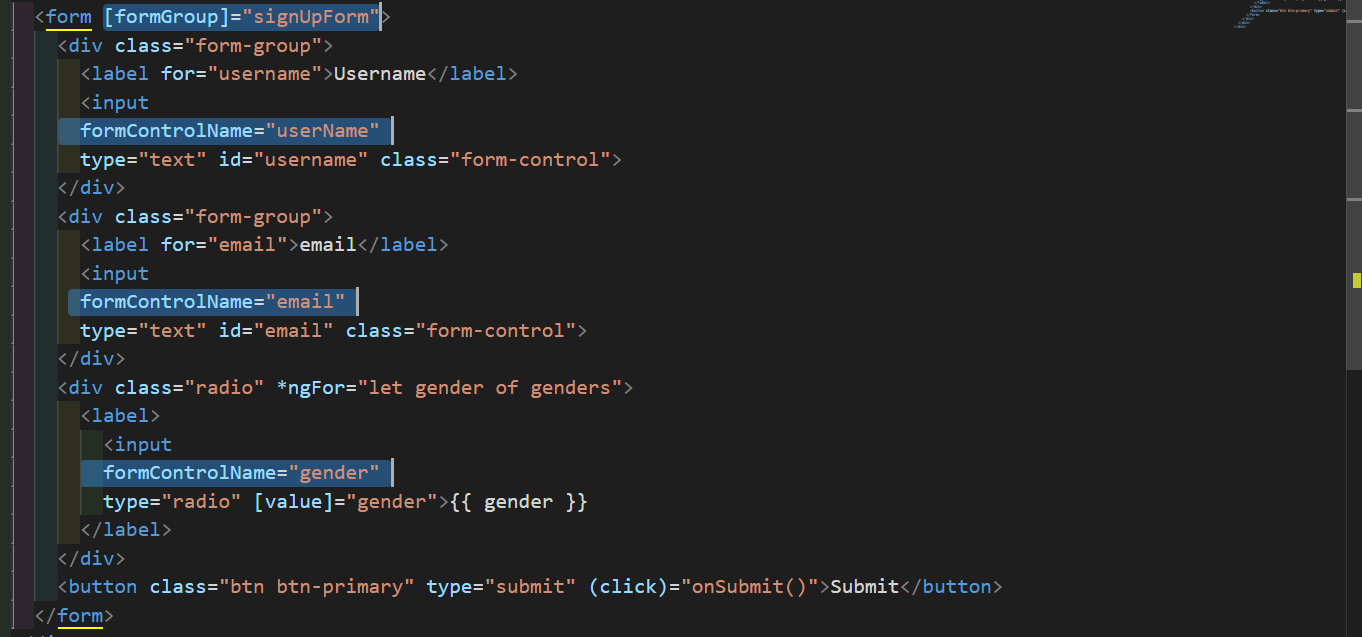
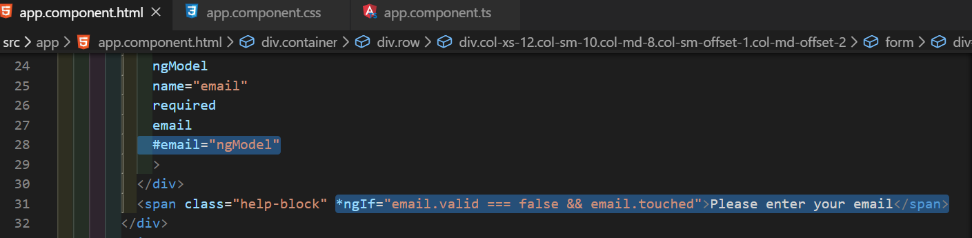
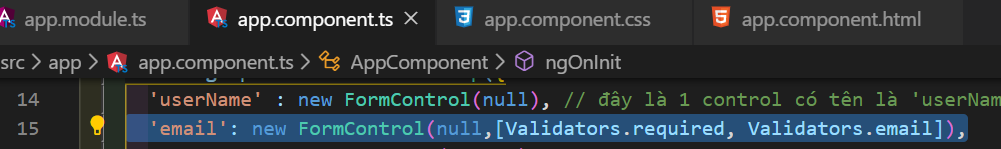
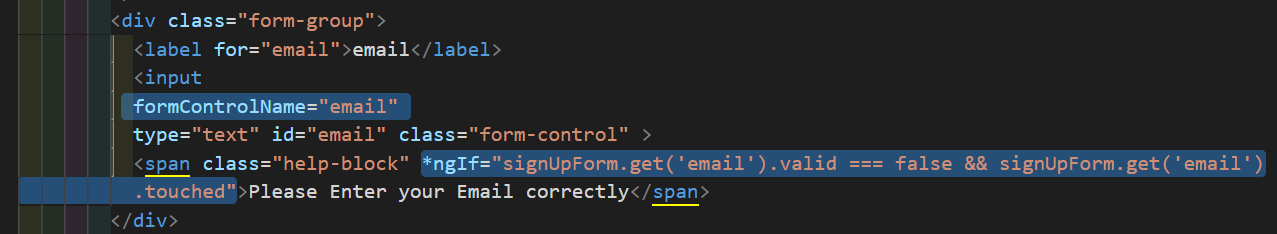
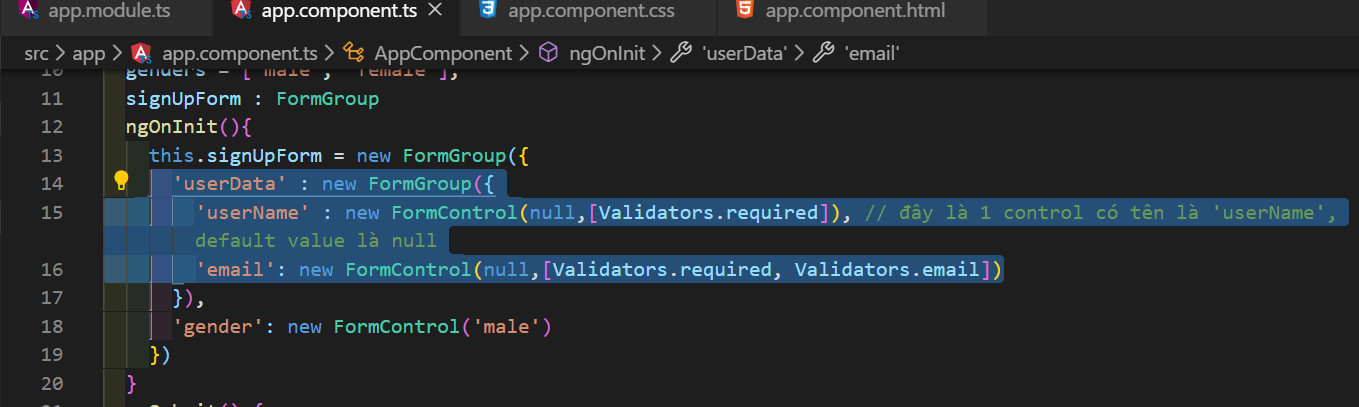
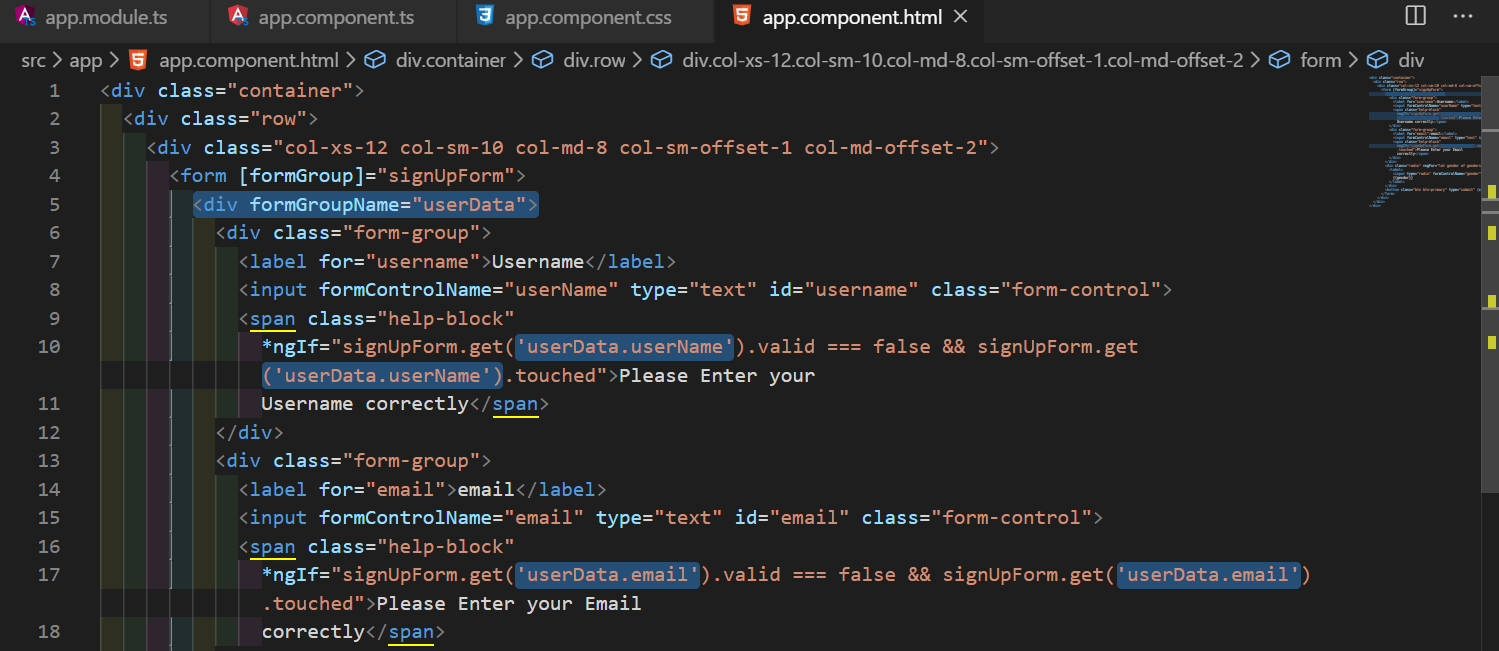
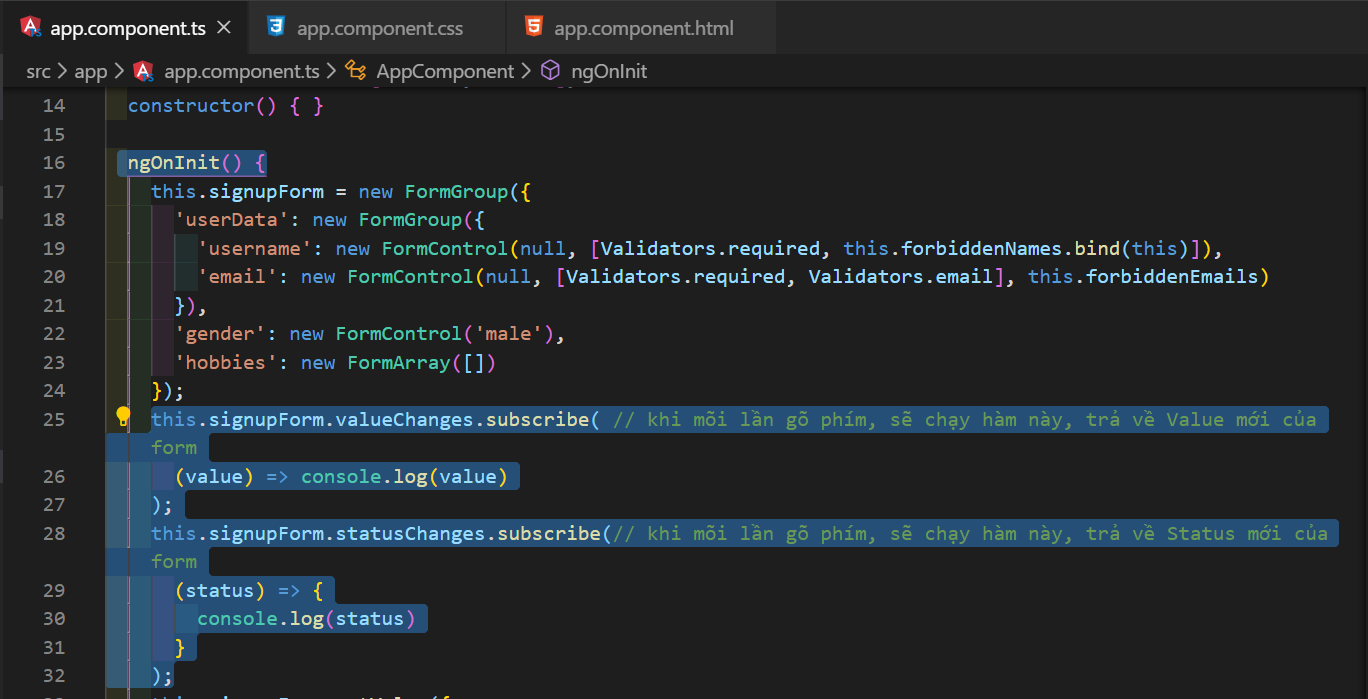
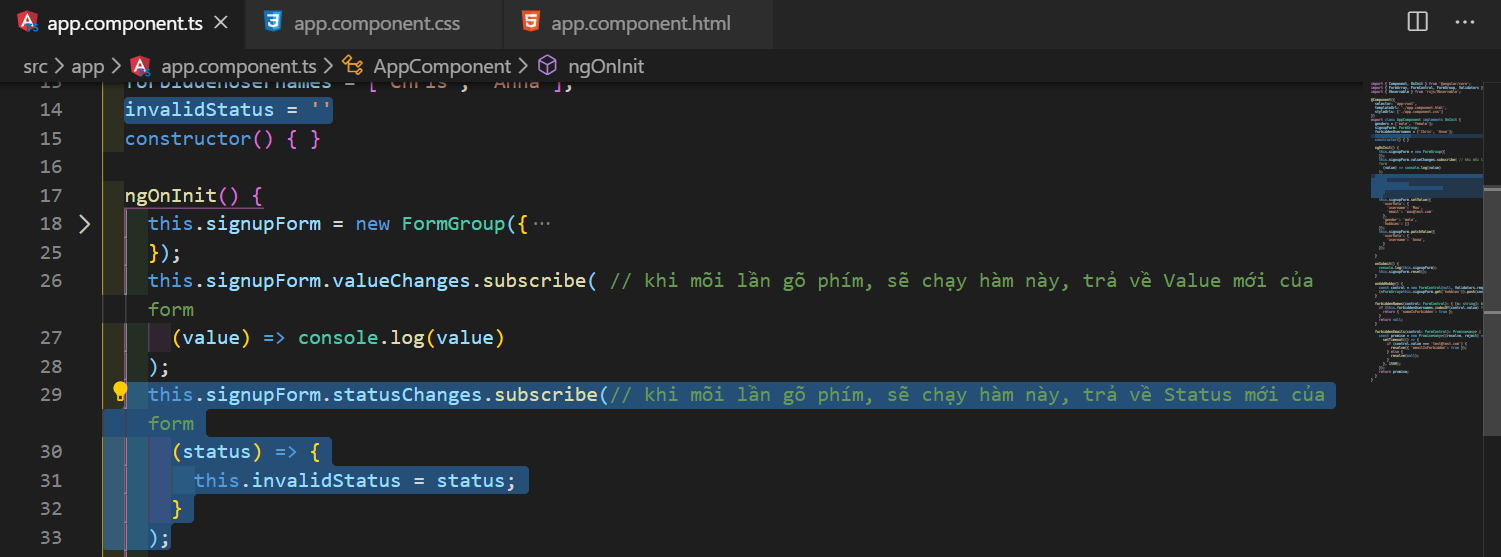
SECTION 15 : **Handling Forms:**

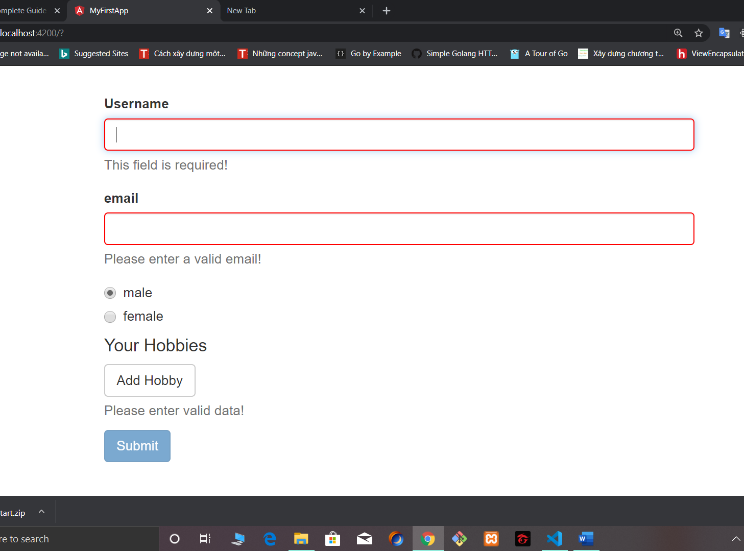
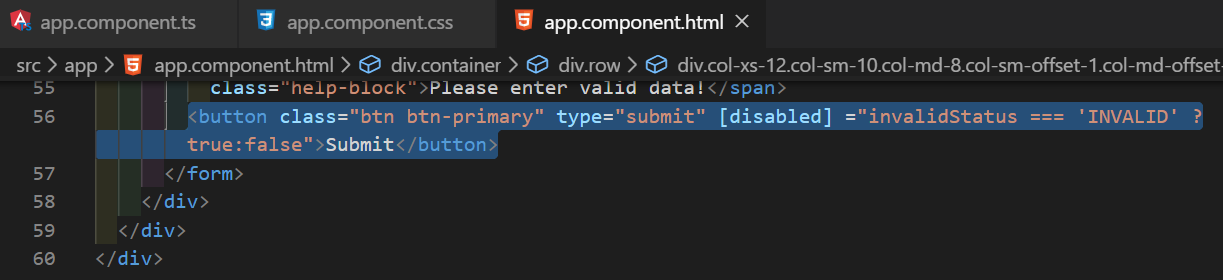
1. **Template-Driven Form:** Tạo Form trong template (trong file HTML): **Phải luôn luôn có property “ngModel” trong các thẻ control thì Form ấy mới quản lý thẻ Control ấy.**
   1. **Creating the Form and Accessing the Controls:** Cách 1: truyền biến Form vào hàm onSubmit(form) trong file HTML
      1. )****
      2. Import FormModule: ****
      3. **-**Tạo 1 cái local Reference #form, truyền vào hàm onSubmit(form).
      4. **-**Khi submit, lấy tất cả giá trị của Form.
   2. **Accessing the Form with ViewChild(‘’):** Cách 2: Truy cập vào Form thông qua ViewChild() trong TypeScript:
      1. ****
      2. **-**Khi dùng cách này thì ta có thể truy cập vào Form bất cứ khi nào, còn cách kia thì chỉ truy cập vào form khi ta click Submit thui.
   3. **Outputting Validation Error Message:** Thực hiện chức năng, khi click vào input email, nhưng không nhập gì, hoặc nhập sai yêu cầu (ví dụ như thiếu @gmail.com ….) thì sẽ xuất ra thông báo
      1. **-**tạo 1 local Reference cho thẻ Input, và xét điều kiện các Properties của thẻ Input để hiển thị ra Warning (\*ngIf).
      2. **-Angular** sẽ dựa vào các Property của thẻ Input mà tự động thêm các Classes này vào, mình có thể vào đây tự động style lại theo ý của mình.(ví dụ như trên là nếu sẽ Input KHÔNG HỢP LỆ && ĐƯỢC CLICK VÀO ).
   4. **Set Default Values with ngModel Property Binding:** Xét giá trị mặc định của các thẻ controls(input,select…) bằng [ngModel]=”DefaultValue”.
      1. ****
      2. ****
      3. ****
   5. **Using ngModel with Two-Way-Binding**
      1. **-**Phải có property ‘name’, value là gì không quan trọng, không có thì sẽ không chạy.
      2. **-**Biến này dùng để chứa dữ liệu của thẻ Textarea kia, khi giá trị tag <textarea> thay đổi thì biến answer cũng thay đổi đồng bộ theo.
   6. **Handling Radio Buttons:** lấy giá trị của Radio button:
      1. ****
      2. ****
   7. **Setting and Patching Form Values:**Khi ấn vào 1 button, giá trị của Form sẽ thay đổi theo giá trị mà mình set trong Typescript:
      1. **-**click vào thì thay đổi giá trị của Form.
      2. **-** Dùng hàm .form.pathValue() để set giá trị của các Controls (input, select) trong Form. Các username, gender,email chính là name của các control ấy (<input name = ‘username’>).
   8. **Resetting Forms:**

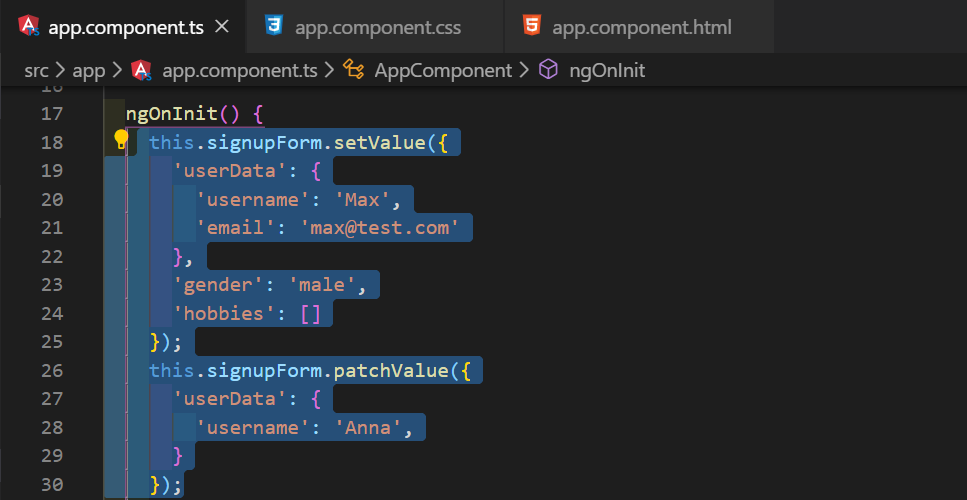
****

**XONG DRIVEN TEMPLATE**

1. **Reactive Approach:**
   1. **Introduction:** 
   2. **Setup:** ****- Import ReactiveFormsModule ở AppModule.ts
   3. **Creating a Form in Code**: tạo 1 cái Form trong template (trong file TypeScript):
      1. ****- Khai báo và import FormGroup vào template mà mình muốn tạo Form.
      2. ****- Khởi tạo Form trong lifecycle hook OnInit().
   4. **Syncing HTML and Form: Đồng bộ hóa, kết nối** Form ở trong template (Typescript) và Form ở trong HTML.
      1. **-** Các **giá trị** của “formGroup”, “formControlName” phải phù hợp với giá trị mình đã khai báo ở Typescript.
   5. **Adding Validation:** Thêm các Validation như Required, email.. bằng code :
      1. **-** Validation là mấy cái email, required của line 26 27
      2. **-**Thêm Required, thêm Email.
   6. **Getting Access to Controls:** Thực hiện chức năng, khi click vào input email, nhưng không nhập gì, hoặc nhập sai yêu cầu (ví dụ như thiếu @gmail.com ….) thì sẽ xuất ra thông báo giống 1.3 nhưng bằng cách khác:
      1. ****
      2. ****
   7. **Grouping Controls:** nhóm các controls (thẻ input username, thẻ input password) lại với nhau để tiện cho việc quản lý Form.
      1. **-** nhóm 2 controls ‘Email’ và ‘userName’ thành ‘userData’, nhóm ‘userData’ là Form con của Form cha ‘signUpForm’.
      2. ** -** Sử dụng formGroupName. Muốn truy xuất userName thì phải truy xuất userData trước.
   8. **Arrays of Form Controls (FormArray).**
   9. **Reacting to Status or Value Changes:**  Theo dõi những giá trị, trạng thái của Form khi thay đổi, từ đó cập nhật lại Giao diện.
      1. ****
      2. **Ví dụ:** từ Form trên, khi nhập không đúng @gmail.com, hoặc không nhập đầy đủ thì nút Submit sẽ bị Disabled (vô hiệu hóa):

** -** Mỗi lần gõ phím, sẽ kiểm tra trạng thái Valid của Form(sẽ trả về biến chuỗi ‘INVALID’ hoặc ‘VALID’ hoặc ‘PENDING’), sau đó gán cho biến invalidStatus để cập nhật lên giao diện.

 **­**- Khi không nhập Username và email, status của form là INVALID, gán ‘INVALID’ cho invalidStatus. Từ đó cập nhật lại giao diện ( nút SUBMIT bị VÔ HIệu HÓA).- Cập nhật lại giao diện từ biến invalidStatus.

* 1. **Setting and Patching Values:** thiết lập giá trị của các controls trong Form:Nên dùng hàm **patchValue()**:
     1.  -hàm setValue() là set cho toàn bộ Form, cái nào ko set thi đưa về mặc định.

- Còn PatchValue() là chỉ set riêng Control đó, còn lại giữ nguyên