**I – Tạo project Angular đầu tiên**

Cài NodeJS và npm, kiểm tra: **node –v** và **npm –v**

1. Download  [Angular CLI](https://github.com/angular/angular-cli):

**npm install -g @angular/cli,** kiểm tra **ng –v**

Nếu có lỗi thì

* Gỡ bỏ angular: **npm uninstall -g @angular/cli**
* xoá thư mục **npm** và **npm-cache** trong **C:\Users\DinhNV\AppData\Roaming**

rồi cài lại với lệnh **npm install -g @angular/cli**

1. Cài yarn: **npm install yarn –g**
2. Tạo thư mục nơi lưu trữ project, và thao tác trong thư mục này
3. Tạo project mới**: ng new my-app**
4. Cd vào project vừa tạo
5. Chạy project lên: **ng serve**
6. Truy cập trình duyệt với đường dẫn **localhost: 4200** để xem project vừa tạo

**II - Tạo component đơn giản**

1. Tạo file word.component.ts, để khai báo component ở trong file này
2. Import component vừa tạo vào file app.module.ts
3. Sử dụng component vừa tạo trong file app.component.html

**III - Component decorators**

Có thể tách riêng các template, style ra thành các file riêng sau đó link vào trong file ts để sử dụng, như vậy code sẽ gọn hơn

1. Tạo các file word.component.css, word.component.html để định nghĩa các thành phần này trong các file riêng biệt
2. Link các file này vào file word.component.ts để sử dụng

**V - Nhúng bootstrap vào project**

npm install --save bootstrap, npm install jquery — save

Thêm đường dẫn bootstrap.min.css vào file angular.json

Thêm đường dẫn jquery.min.js vào file angular.json

**VI – Truyền giá trị vào trong html:**

* Chỉnh sửa trong file word.component.ts để đặt giá trị cho các biến
* Vào file word.component.html để sử dụng các biến vừa định nghĩa

**VIII - Tạo component sử dụng Angular CLI**

1. Mở cửa sổ terminal mới vào sử dụng cú pháp: **ng g c tên\_component**
2. Khi đó một component được tạo ra và tự động import vào file app.module.ts
3. Vào file app.component.html để sử dụng component vừa tạo

# IX-  Property binding (tạo ra các biến và sử dụng – ràng buộc thuộc tính)

# Vào file word.component.ts để định nghĩa các biến

# Vào file word.component.html để sử dụng các biến vừa định nghĩa

# thuộc tính phải để trong dấu ngoặc vuông

# X - Event binding (tạo ra các biến và sử dụng – ràng buộc sự kiện)

# Tạo một sự kiện trong file word.component.html

# (Sự kiện phải để trong dấu ngoặc tròn)

# Định nghĩa các hàm trong file word.component.ts và

# sử dụng khi tác động vào sự kiện

# Bài 11: Xử lý text input

# Tạo một component mới: ng g c textinput

# Sử dụng component vừa tạo trong file app.component.html

# Trong folder component vừa tạo, trong file html tạo 1 thẻ input để nhập username gồm thuộc tính placeholder và sự kiện keyup, đồng thời hiện giá trị vừa nhập vào từ ô input

# Trong file textinput.component.ts, định nghĩa một hàm showEvent để thực hiện chức năng khi nhập vào ô input thì sẽ hiển thị ra ở file html

# Trong file html, khi nhập vào ô input thì gọi hàm showEvent với tham số truyền vào là $event

# Bài 12: Two way binding

# Import thư viện vào file app.module.ts để sử dụng được thuộc tính [(ngModel)]

# import { FormsModule } from '@angular/forms';

# FormsModule

# Tạo component mới tên là waybinding tương tự như textinput

# Tương tự như xử lý textinput, nhưng ở trong file html không có sự kiện keyup mà thay bằng sự kiện [(ngModel)], và không cần định nghĩa hàm ở trong file …component.ts luôn

# Bài 13: Template reference variable

# Cũng trong component waybinding, ở file html , Thêm biến #txtUsername vào thẻ input,

# Hiển thị kết quả của biến txtUsername ở trong file form.component.html luôn

# Bài 14: Style binding và class binding

# Tạo component mới tên là styleclassbinding tương tự như waybinding ở trên

# Trong file html, thêm thuộc tính [style.color] = “name.length%2==0? ‘red’ : ‘black’”

# Để khi nhập vào số ký tự là chẳn thì hiển thị chữ màu đỏ, nếu không thì hiển thị chữ màu đen

# Có thể thêm biến isHighLine = true ở file ts và sử dụng trong file html

# Khi isHighLight có giá trị true thì chữ màu đỏ và ngược lại

# Thêm thẻ div trong file html có class là circle hoặc square, để sử dụng class binding

# Bài 15:Cơ bản về directive,ngStyle,ngClass

# Tạo một component mới có tên là directive

# Thêm thuộc tính [ngStyle] vào thẻ p trong file html

# Chỉnh sửa trong file ts, để khi biến isHighLight có giá trị true thì hiển thị chữ màu đỏ,

# Ngược lại thì hiển thị chữ màu xanh,

# Thuộc tính [ngStyle] dùng để định nghĩa nhiều thuộc tính [style] binding

# Thêm thuộc tính [ngClass] vào thẻ div trong file html

# Chỉnh sửa trong file ts, để khi biến isHighLight có giá trị true thì hiển thị hình vuông,

# Ngược lại thì hiển thị hình tròn

# Bài 16: ngIf

# Tạo component mới có tên là struct

# Thêm thuộc tính \*ngIf=”isShow” vào thẻ p, đồng thời thêm button Toggle để khi kích vào thì thực hiện hàm getShow()

# Trong file ts, định nghĩa hàm getShow() để khi biến isShow có giá trị true thì hiển thị thẻ p và ngược lại

# Bài 17: ngFor đơn giản

# Cũng trong component struct, tạo thêm một mảng arrSubjects trong file ts, là mảng chứa các môn học

# Trong file html, tạo một list để hiện thị danh sách các môn học,

# sử dụng vòng lặp \*ngFor=”let subject of arrSubjects”

# Bài 18: ngFor với mảng đối tượng

# Tạo project Angular có tên Bai3NgModel

# Tạo component mới có tên words

# Trong file ts, tạo một mảng arrWords để chứa các đối tượng từ vựng

# với các thuộc tính: *id, en, vn, memorized*

# 4. Trong file html, tạo ra thẻ div, dùng vòng lặp \*ngFor=”let word of arrWords” để hiển thị danh sách các từ vựng sử dụng tất cả các thuộc tính, bên cạnh đó cũng sử dụng [ngStyle] để đặt màu cho thẻ h4Bài 19: Thêm phần tử vào mảng kết hợp ngFor

# Tạo component mới có tên addword, tương tự component words ở trên

# Trong file html, thêm 2 thẻ input để có thể nhập vào từ tiếng Anh và nghĩa tiếng Việt

# Cũng trong file này, thêm một button để khi kích vào thì thực hiện thêm một từ mới vào

# Trong file ts, thêm các biến newEn, newVn để chứa giá trị của các biến nhập vào từ file html

# Cũng trong file này, định nghĩa hàm addWord() để khi kích vào button ở file html thì thực hiện thêm một từ mới vào, đồng thời xóa trắng ô input ban đầu

# Bài 20: Ẩn hiện form

# Tạo project Angular mới tên Bai4Form

# Tạo component mới có tên là showform, tương tự như component addword ở trên

# Trong file html: Tạo thẻ div có thuộc tính \*ngIf=”isShowForm” chứa 2 thẻ input và thẻ button thêm,

# Đồng thời tạo thêm một thẻ input Thêm từ mới2 ở ngoài thẻ div, để khi nhấn vào thì hiện div input ra

# Trong file ts: tạo biến isShowForm=false; để hiện button Thêm từ mới2 mà không hiện form input,

# Khi nhấn vào button Thêm từ mới2, hiện form input ra và ẩn button Thêm từ mới2 đi

# Bài 21: Xóa phần tử trong mảng hiển thị

# Tạo component có tên là delete, tương tự như conponent words ở trên

# Trong file html, tạo thuộc tính class=”word” để định dạng cho các word trong giao diện

# (Nhớ định nghĩa class word ở trong file css)

# Đồng thời thêm button để khi kích vào thực hiện hàm xóa từ

# Trong file ts, định nghĩa hàm removeWord(id) để xóa một từ có chỉ số là index,

# Trong đó index là chỉ số của từ cần xóa

# Bài 22: Lọc mảng hiển thị

# Tạo component có tên filter, tương tự như component words ở trên

# Trong file html, Thêm thẻ select để hiển thị các lựa chọn hiển thị, biến filterStatus để lưu giá trị của thẻ select

# Thêm thuộc tính \*ngIf để hiển thị các từ với các điều kiện

# Nếu giá trị của biến filterStatus = “tat\_ca” thì hiển thị tất cả các từ

# Nếu giá trị của biến filterStatus =”da\_nho” và word.memorized = true thì hiển thị ra (các từ đã nhớ)

# Nếu giá trị của biến filterStatus =”chua\_nho” và word.memorized = false thì hiển thị ra (các từ chưa nhớ)

# Có thể Tách riêng các điều kiện vào hàm getShowStatus() trong file ts rồi sau đó

# sử dụng hàm đó trong thuộc tính \*ngIf trong file html

# Bài 23: Thay đổi trạng thái phần tử trong mảng hiển thị

# Tạo component có tên change, tương tự như component delete ở trên

# Thêm button vào file html, để khi kích vào thì thay đổi giá trị của thuộc tính trong đối tượng

# Thêm biểu thức để biết trạng thái của word

# Bài 24: Input

# Tạo project Angular mới có tên Bai5Input

# Tạo component mới có tên person

# Sử dụng component vừa tạo trong file app.component.html, khai báo thêm các biến với giá trị tùy ý

# Trong file component.ts, import thêm thư viện Input vào,

# đồng thời nhận giá trị các biến bằng dòng lệnh @Input() tên\_biến: kiểu\_dữ\_liệu;

# Sử dụng các biến đó trong file html, với cú pháp {{ tên\_biên }}

# Bài 25: ngFor kết hợp Input

# Tạo component mới có tên listperson

# Trong file listperson.component.ts, tạo ra một mảng arrPeople

# để chứa các đối tượng người có thuộc tính name và age

# Trong file listperson.component.html, dùng \*ngFor để hiện tất cả đối tượng của mảng arrPeople,

# Sử dụng component person vừa tạo ở bài trên để truyền biến name và age vào

# Sử dụng component listperson trong file app.component.html để hiện kết quả ra

# Bài 26: Output 1

# Tạo ra file parrent.component.ts, và định nghĩa thù công một component, nhớ import vào file app.module.ts

# Sử dụng parrent.component vừa định nghĩa trong file app.component.html

# Tạo ra file child.component.ts và định nghĩa một component, cũng nhớ import vào file app.module.ts

# Sử dụng child.component.ts vào file parrent.component.ts

# Bài 27: Output 2

# Cũng trong parrent.component.ts vừa tạo ở bài trên, thêm sự kiện myClick vào thẻ <app-child> để khi kích vào thì tăng giá trị của biến value lên 1 (Sự kiện myClick do mình tự đặt tên)

# ở file child.component.ts, import thêm thư viện Output và EventEmitter vào

# đồng thời nhận giá trị của sự kiện myClick = new EventEmitter();

# cũng trong file này, ở thẻ button trên thuộc tính template, thêm sự kiện (click) để khi kích vào thì thực hiện hàm addForParrent()

# cũng trong file này, định nghĩa hàm addForParrent(){

# this.myClick.emit();

# }

# Bài 28: Ouput có tham số

# Trong file parrent.component.ts, thay đổi sự kiện myClick để khi kích vào thì thực hiện hàm changeValue(),

# Cũng trong file này, định nghĩa hàm changeValue() để thay đổi giá trị của biến value

# (tham số truyền vào hàm này là giá trị true hoặc false)

# Trong file child.component.ts, thêm 1 thẻ button nữa để khi kích vào thì thực hiện hàm subForParrent()

# Cũng trong file này, định nghĩa hàm subForParrent tương tự như trên, nhưng có tham số ở emit là true hoặc false,

# Trong file parrent.component.ts, thay đổi tham số của hàm changeValue($event),

# Lúc này, thay vì truyền tham số true, false vào hàm changeValue() thì ta truyền true, false vào emit

# Để emit chỉ nhận giá trị true hoặc false thôi thì thêm kiểu này vào hàm EventEmitter<boolean>()

# Bài: 29 Bài tập Output

# Tạo component person và listperson, có sử dụng input để hiển thị danh sách người giống bài trước

# Trong file listperson.component.html, thêm sự kiện (myClick) để khi kich vào thì thực hiện hàm removePerson(); tham số truyền vào khi kích là $event

# Trong file listperson.component.ts, định nghĩa hàm removePerson() để xóa phần từ mà file html yêu cầu

# Trong file person.component.html, thêm sự kiện (click)=”removeByClick()”

# Trong file person.component.ts, nhận sự kiện (myClick), khai báo hàm removeByClick()

# Làm thêm nút Tăng tuổi để khi kích vào thì thực hiện tăng thêm một tuổi

# Bài 30: Viewchild 1

# Sử dụng lai component parrent và child của bài trước

# Chỉnh sửa lại file child.component để thẻ h3 hiện giá trị biến value

# Trong file parrent.component, hiện nút button Add for child, đồng thời render app-child vào

# Bài 31: ViewChild 2

# Import component ChildComponent vào parrent.component

# Trong class ParrentComponent

# sử dụng @ViewChild(ChildComponent)

# Khai báo biến myChild có kiểu ChildComponent

# Định nghĩa hàm onAddForChild() để tăng giá trị biến myChild lên 1

# Thêm sự kiện (click) vào button để khi kích vào thì gọi hàm onAddForChild()

# Bài 32: ngContent

# Tạo một component mới có tên là card.component.ts

# Trong component này, hiển thị 1 thẻ div có class là card chứa 2 thẻ div có class là header và body

# Trong các div này, hiển thị một thẻ ng-content có các class được khai báo ở file app.component.html

# Định nghĩa class card ở trong file card.component.ts luôn

# Bài 33: Build in pipes

# Tạo component mới có tên là learn-pipe

# Trong file ts, khai báo các biến với các kiểu dữ liệu Date, json, String

# Trong file html, hiển thị các biến đã khai báo trong file ts, nhớ sử dụng pipe để hiện định dạng hiển thị cho đẹp, dễ hiểu

# Bài 34: Custom pipe

# Tạo pipe mới có tên round.pipe.ts

# Import pipe này vào trong app.module.ts

# Định nghĩa pipe này…

# Sử dụng pipe trong file learn-component.html

# Bài 35: Khái niệm service và http service

# Tạo một folder Angular mới tên là Bai8httpservice

# Trong folder này, tạo thủ công một component mới có tên ip.component.ts

# Import component này vào file app.module.ts, đồng thời cũng import thư viện HttpModule vào file module này luôn

# Trong component mới tạo này, định nghĩa một phương thức constructor để get dữ liệu từ server về, đồng thời hiện kết quả ra để kiểm tra (nhớ import thư viện Http vào…)

# Bài 36: Tách service

# Tạo file ip.service.ts để định nghĩa phương thức getIp(), import thư viện Injectable

# Import file này vào app.module.ts

# Tạo file ip.component\_service.ts để sử dụng phương thức getIp() đã được định nghĩa,

# Trong file này, import ip.service.ts vào, đồng thời cũng thêm providers: [IpService] vào

# Bài 37: Sử dụng ngOnInit

# Tạo file component mới có tên ip.component\_ngOnInit.ts, sử dụng phương thức getIp(), nhớ import thư viện OnInit, và class phải implements OnInit,

# Trong class IpComponentngOnInit, phải có hàm ngOnInit()

# Import component này vào file app.module.ts

# Có thể tách thuộc tính providers: [IpService] vào file app.module.ts để sử dụng cho gọn code

# Hàm ngOnInit() sẽ tự động chạy trong class, còn các hàm khác chỉ chạy khi gọi đến

# Bài 40: Weather service

# Tạo component mới có tên weather.component.ts

# Trong component này, tạo file weather.service.ts, trong file này, định nghĩa hàm getTemp() để lấy thông tin nhiệt độ từ server

# Trong file weather.component.ts, gọi hàm getTemp() đã định nghĩa để lấy thông tin nhiệt độ từ một thành phố, tham số truyền vào là thành phố cần lấy nhiệt độ, log nhiệt độ ra console để kiểm tra, nhớ thêm thuộc tính provoders

# Bài 41 Lấy nhiệt độ và hiển thị

# Trong component weather, chỉnh sửa file html để hiển thị nhiệt độ, form nhập vào một thành phố, đồng thời hiển thị button để khi kích vào thì hiển thị nhiệt độ cần lấy

# Trong file ts, định nghĩa hàm getWeather() để lấy nhiệt độ, khi nhập thành phố vào ô input trong file html

# Sau khi lấy nhiệt độ và hiển thị nhiệt độ thì xóa sạch ô input

# Bài 42: Tính năng loading và xử lý lỗi

# Cũng trong component weather, trong file ts, khai báo thêm biến isLoading = false;

# Mỗi khi gọi hàm getWeather(), gán biến isLoading = true, sau khi get được dữ liệu xong, gán lại biến isLoading = false;

# Định nghĩa hàm getMessage() để hiển thị dữ liệu ra file html, khi isLoading = true thì return “Loading…”

# Bài 43: Giới thiệu về form

# Tạo một Angular mới có tên là Bai9Form

# Tạo một component có tên là sign-in.component.ts

# Trong component này, tạo 2 thẻ input để nhập email và password, và một button Submit

# Khi nhấn vào button này thì thực hiện log ra email và password vừa nhập vào

# Bài 44: ngForm

# Tạo component mới ng g c ng-form

# Trong file html, tạo form có 2 thẻ input để nhập vào email và password, mỗi thẻ input có thêm thuộc tính [(ngModel)], và phải có thuộc tính name, có thêm button để khi click thì thực hiện submit form

# Cũng trong file này, khi nhấn nút Submit thì sự kiện (submit) của form gọi hàm onSubmit(formSignIn), tham số #formSignIn là một reference variable, gán bằng ngForm

# Trong file ts, định nghĩa hàm onSubmit(formSignIn) để log ra giá trị formSignIn.value

# Bài 45: Validate template form

# Có thể thêm thuộc tính required vào các thẻ input, khi đó nếu chưa nhập thông tin thì thuộc tính valid của các thẻ này là false, và do đó thuộc tính valid của form cũng là false

# Có thể thêm đoạn lệnh

# <p \*ngIf="formSignIn.controls.email?.errors.required">

# Email is required

# </p>

# Vào file html để nếu chưa nhập vào ô input thì hiện thông báo lỗi

# Bài 46: ngSubmit

# các thuộc tính của ngForm thường đi theo cặp, và có giá trị đối nhau

# dirty: true ⬄ pristine: false ô input đã nhập chưa?

* 1. touched: true ⬄ untouched: false: đã chạm vào ô này chưa?

# valid: true ⬄ invalid: false: đã valid ô này chưa?

# submitted: true ⬄ pedding: false: đã submit chưa?

# Có thể xóa các dấu [()] của ngModel đi, tức là không cần gán giá trị cho các ô input

# Khi thực hiện throw new Error("Form is invalid!"); để ném ngoại lệ, thì giá trị của form sẽ được submit đi

# Sửa (submit) form thành (ngSubmit) để khi thêm câu lệnh throw ngoại lệ ở file ts thì form chỉ ném ra ngoại lệ ở console.log mà không cho submit, tức là không tải lại trang

# Bài 48: Xử lý style các input không hợp lệ

# Để tránh viết dài dòng thì có thể khai báo thêm biến #txtEmail = “ngModel” vào ô input,

# Khi đó, có thể thay thuộc tính formSignIn.controls.email? bằng giá trị của biến txtEmail

# Ngoài thuộc tính required, có thể thêm các thuộc tính valid khác như email,… vào thẻ input

# Khi đó muốn hiểu thị tất các lỗi có thể thêm câu lệnh <p>{{ txtEmail.errors | json }}</p> vào file html

# Nếu không muốn hiển thị câu thông báo valid ngay khi tải trình duyệt thì có thể thêm câu lệnh này vào \*ngIf="txtEmail.touched && txtEmail.errors?.required"

# Có thể vào trình duyệt -> kiểm tra nguồn trang, để biết class các phần tử, dựa vào class để định dạng trang cho đẹp, viết css vào file style.css

# Bài 49: Xử lý checkbox và ngModelGroup

# Có thể thêm các thẻ checkbox vào file html

# Có thể gom nhóm các thẻ input với thuộc tính ngModelGroup=”subject”

# Thêm biến #txtPassword="ngModel" để gán giá trị ô password cho biến txtPassword

# Có thể dùng dòng lệnh này để hiện thị những validate

# xem cho biết <p>{{ txtPassword.errors | json }}</p>

# Ngoài validate required và email, còn có các validate khác như minlength, maxlength, pattern,

# Bài 50: Tạo server nodejs

# tạo file Angular mới có tên Bai10Server ,

# Tạo thư mục server chứa server, và cd vào thư mục này để thao tác với server

# Tạo ra project mới npm init –y => Tạo ra file package.json

# Cài đặt express: npm install express --save

# Tạo file index.js để viết code tạo ra server express

# Để chạy server, dùng lệnh *node index*, vào trình duyệt localhost: 3000 để xem kết quả

# Sử dụng phương thức post để post thông tin lên server, dùng công cụ postman

# Sử dụng thư viện body-parser để nhận các thông tin khi gửi lên

# cài body-parser: npm install body-parser --save

# thêm vài dòng lệnh để cấu hình lại server, cho việc gửi dữ liệu lên

# cấu hình lại postman cho việc gửi dữ liệu lên

# Bài 51: Gửi request từ form angular

# copy Component ng-form của bài trước vào Angular Bai10Server

# trong file ts, import thêm thư viện http, Headers vào,

# trong file html, thêm thẻ button để khi click vào thì thực hiện hàm postToExpress()

# định nghĩa hàm postToExpress() ở file js, có

# url là đường dẫn đến localhost:3000/signin

# khởi tạo các thành phần khác để thực hiện post dữ liệu lên server

# log kết quả ra console để kiểm tra

# có thể post 1 form lên server bằng cách thay đổi thông tin trong thuộc tính JSON.stringify()

# Bài 52: Viết service gửi request post

# Tạo file ng-form.service.ts để viết code service, định nghĩa hàm sendPost()

# Copy file ng-form.component.ts, chình sửa thành file ng-form.component\_service.ts

# Sau đó gọi hàm sendPost() đã định nghĩa,

# Kết quả nhận được vẫn tương tự như trên

# Bài 53: Reactive form

# Tạo folder Angular mới có tên Bai11Reactive

# Tạo component mới có tên signup.component.ts

# Import ReactiveFormModule vào app.module.ts

# Trong file ts,

# import FormGroup và FormControl vào

# Khai báo hàm constructor() để lấy giá trị của form

# Trong file html,

# gán [formGroup] = “formSignUp” cho thẻ form

# gán formControlName cho các thẻ input

# Bài 54: Nested form group

# thêm các thẻ checkbox vào file html, đặt formControlName cho các thẻ checkbox

# nhóm các thẻ checkbox lại và gán thuộc tính formGroupName

# gán các biến của thẻ checkbox trong constructor trong file ts

# Bài 55: Form builder

# tạo component mới đặt tên signup2.component.ts, từ component signUp copy sang

# chỉnh sửa lại hàm constructor, khai báo thêm hàm ngOnInit() và lấy giá trị của form trong hàm ngOnInit() này

# import thư viện FormBuilder vào

# Khai báo biến FormBuilder ở trong hàm constructor

# Sử dụng các biến FormBuilder trong hàm ngOnInit() chương trình sẽ ngắn gọn hơn,

# Có thể gán trước các giá trị cho các biến trong file ts

# Bài 56: Reactive form validator

# Trong file ts

# Import thư viện Validators

# Trong hàm ngOnInit(), chỉnh sửa lại giá trị của biến email, để validator biến này là yêu cầu nhập

# Thêm thuộc tinh disabled để khi formSignUp,invalid thì ẩn nút Submit đi

# Chỉnh sửa lại file style.css, để khi ô input có class ng.invalid và ng.touched thì hiển thị border màu đỏ

# Thêm thẻ p dưới thẻ input để khi chưa nhập vào thì in ra câu: Email is required

# Bài 57: Custom reactive form validator

# định nghĩa 1 function validator ở trong file ts, có tên là gmailValidator()

# sử dụng validator đã định nghĩa ở trong biến email, để validator yêu cầu đúng định dạng gmail cho biến này

# trong file html, hiển thị các lỗi của input email ra để xem cho biết

# một biến có thể sử dụng nhiều validator bằng cách để các validator trong dấu ngoặc vuông, cách nhau bời dấu phẩy

# Bài 62: Component modules

# Tạo ra các module trong mỗi component, để sử dụng

# Khi đó chỉ cần import module này vào app.module thôi, không cần phải định nghĩa ở trong app.module nữa

# Bài 63: Deploy github page