# **คู่มือ IO-Stagram**

**-------------------------------------------------------------------------------------------**

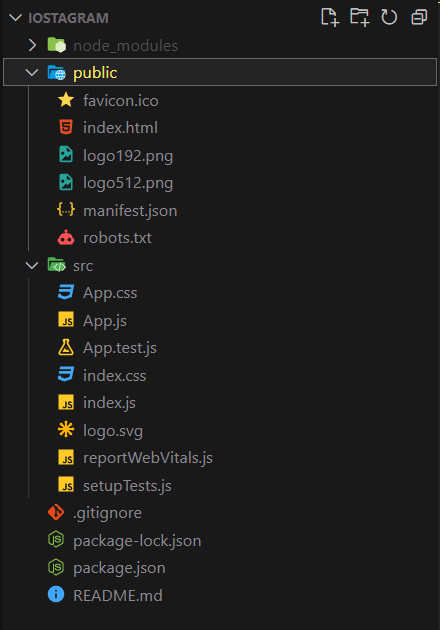
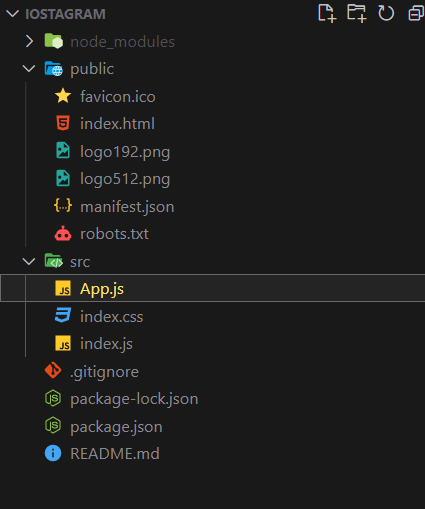
## **ขั้นตอนการเตรียมสภาพแวดล้อม**

1. ติดตั้ง npm ไว้ในเครื่องให้เรียบร้อย เข้าไปดาวน์โหลดได้ลิงค์นี้ <https://www.npmjs.com/get-npm>

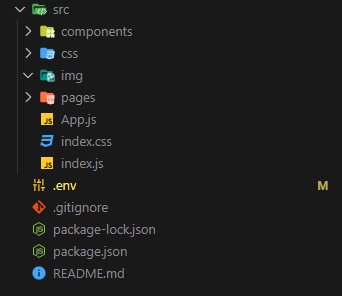
2. สร้างโปรเจ็กต์เริ่มต้นขึ้นมาจากเครื่องมือที่ชื่อ create-react-app ที่จะสร้างไฟล์พื้นฐานต่างๆที่จำเป็นในการเขียน React ขึ้นมา ให้ทำการติดตั้งด้วยคำสั่ง npm install -g create-react-app

3. สร้างโปรเจ็กต์ใหม่ขึ้นมาด้วยคำสั่ง npx create-react-app ชื่อโปรเจ็กต์

4. ทำการลบไฟล์ที่ไม่จำเป็น และลบ code ส่วนที่ไม่ได้ใช้ด้วย

5. สร้างโฟลเดอร์ใหม่อีก 3 โฟลเดอร์ที่จำเป็น คือ component , page , css ไว้ใน โฟลเดอร์ src



**-------------------------------------------------------------------------------------------**

## **เริ่มสร้างโปรเจ็กต์**

### **1. ลง package ที่จำเป็นต้องใช้**

#### 1.1 package สำหรับการทำงาน

- react-router-dom สำหรับการจัดการ route

- axios สำหรับการจัดการกับ api

- dotenv สำหรับเรียกใช้ไฟล์ .env

#### 1.2 package สำหรับการตกแต่ง

- reactstrap สำหรับใช้ component ของ bootstrap

- react-bootstrap สำหรับใช้ component ของ bootstrap

- react-icons สำหรับใช้ icon ต่างๆ

### **2. จัดการ Route สำหรับหน้าต่างๆ ในไฟล์ App.js**

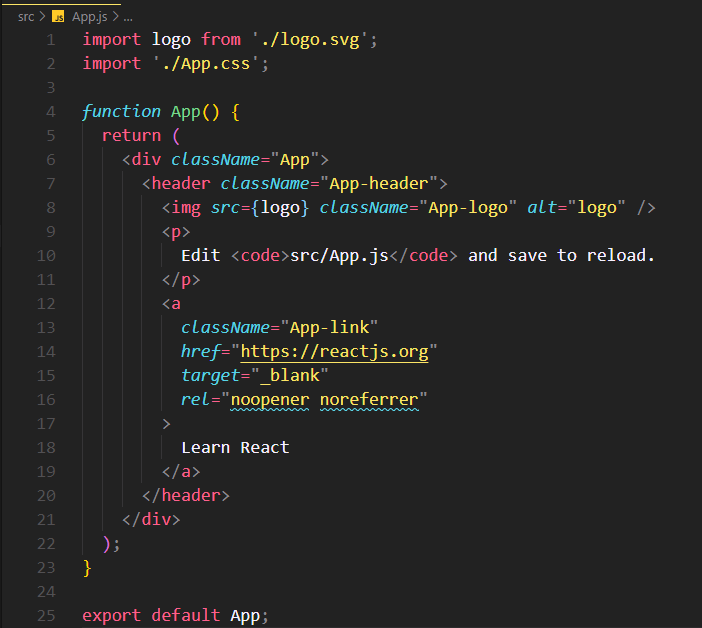
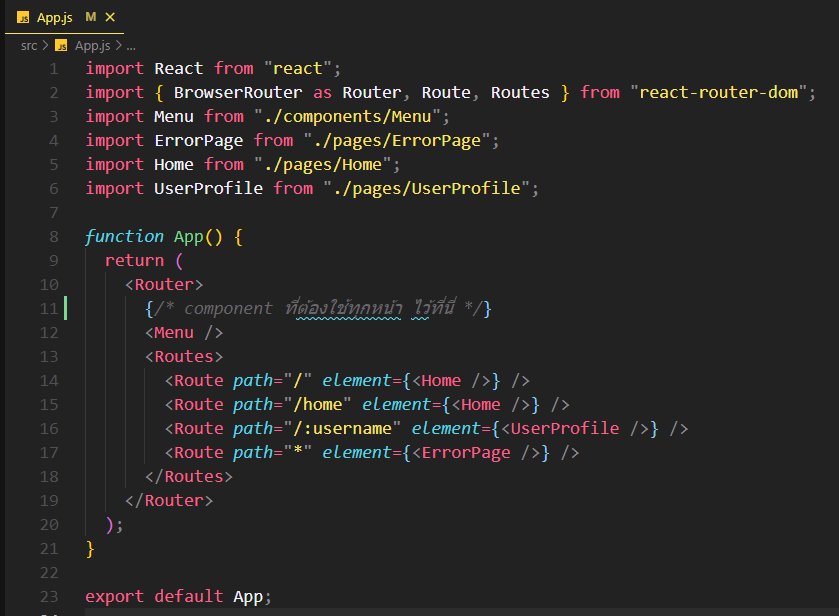
#### 2.1 import package ที่จำเป็นเพิ่มเติมคือ react-router-dom

#### 2.2 import component ของหน้าต่างๆที่อยู่ในโฟลเดอร์ page

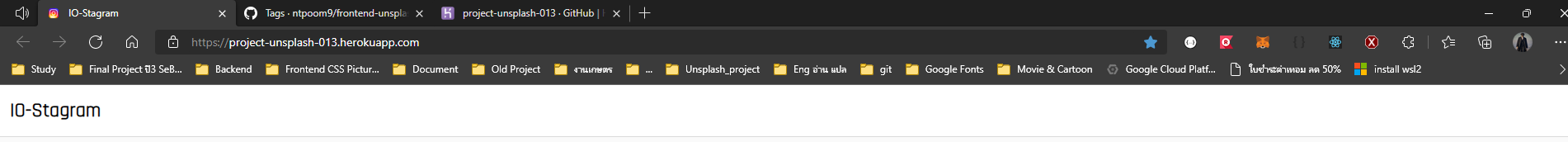
#### 2.3 วางโครงสร้างของ Router

ในแท็ก Route มี attributes อยู่ 2 ตัวคือ  
 - path คือ url หรือ path บนเบราว์เซอร์

- element คือ สำหรับเรียก component หน้าไหนมาใช้ และในหน้านั้นๆก็จะมี component ของตัวเองอีกที

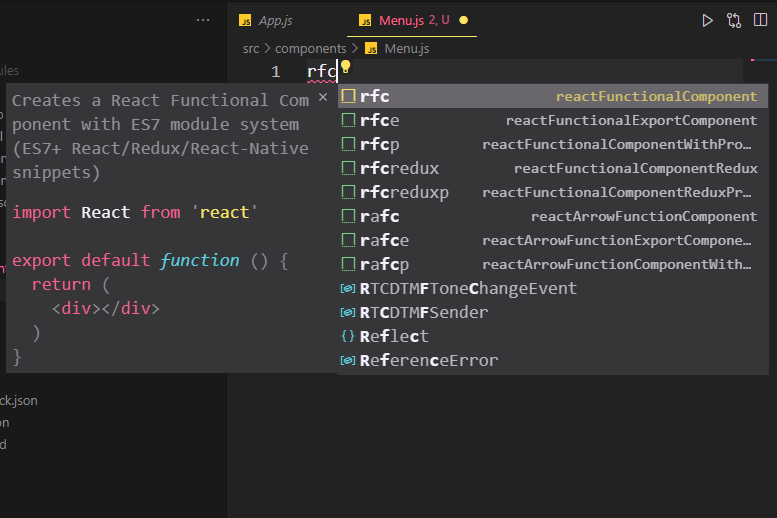
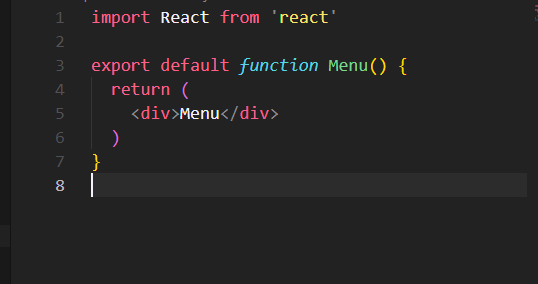
### **3. เริ่มสร้าง component Menu หรือ Navbar**



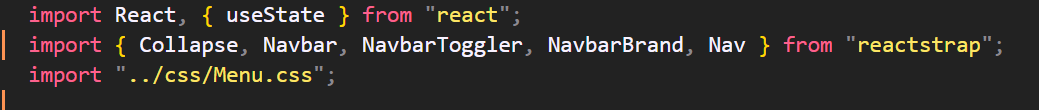
#### 3.1 สร้างไฟล์ Menu.js ในโฟลเดอร์ components



#### 3.2 สร้างเป็น reactFunctionComponent พิมพ์ตัวย่อได้ 2 แบบ rfc rfce จะได้โค้ดเริ่มต้นตามนี้

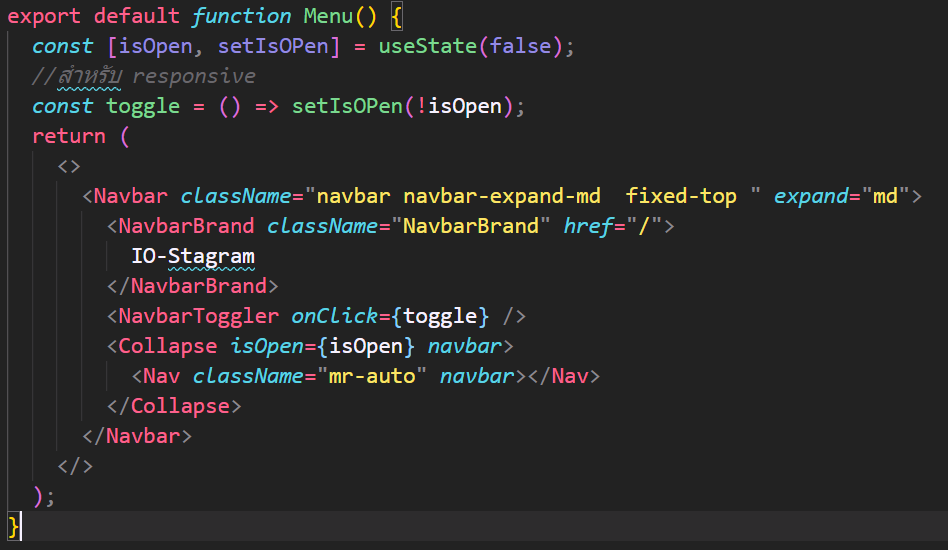
 

#### 3.3 import package เพิ่มเติมที่ต้องใช้



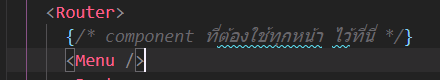
#### 3.4 ใช้ useState มาช่วยในการทำ NavbarToggle กรณีมีเมนูเยอะ แต่ในตัวอย่างมีแค่การค้นหาอย่างเดียว

#### 3.5 ในส่วนฟังก์ชั่นที่ return โครงสร้างออกมา ให้เรา เปิดปิดแท็กอะไรก็ได้ไว้ก่อน เช่น <> , <div> เพราะเป็นโครงสร้างของ JSX ที่ต้องใช้กับ React

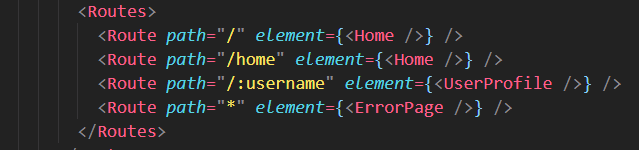


### **4. การนำ component ไปใช้งาน**

#### 4.1 component แบบที่ใช้ซ้ำบ่อยๆ หรือ ต้องมีในทุกๆหน้า เช่น Menu component จะถูกเรียกไว้ในไฟล์ App.js ในส่วนของ component Router



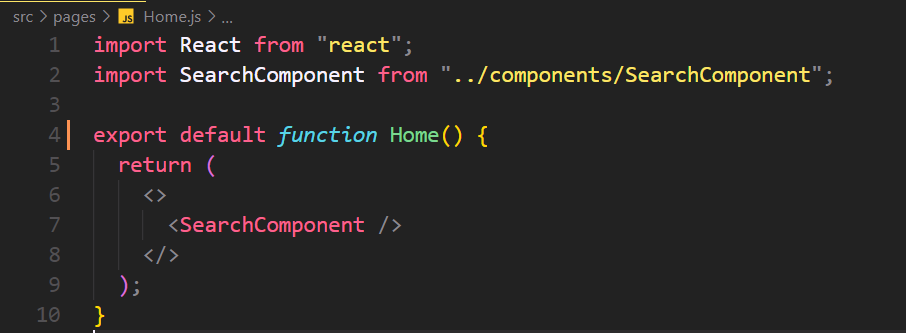
#### 4.2 component ที่เป็นองค์ประกอบเฉพาะหน้านั้นๆ ซึ่งจะแตกต่างกันไป เช่น Search component จะถูกเรียกไว้ในไฟล์ Home.js แทน เพราะเป็นหน้าแรกที่ต้องใช้ Search component ส่วนหน้า Home component จะถูกเรียกไว้ในไฟล์ App.js แทน ในส่วนของแท็ก Routes ใน attributes element



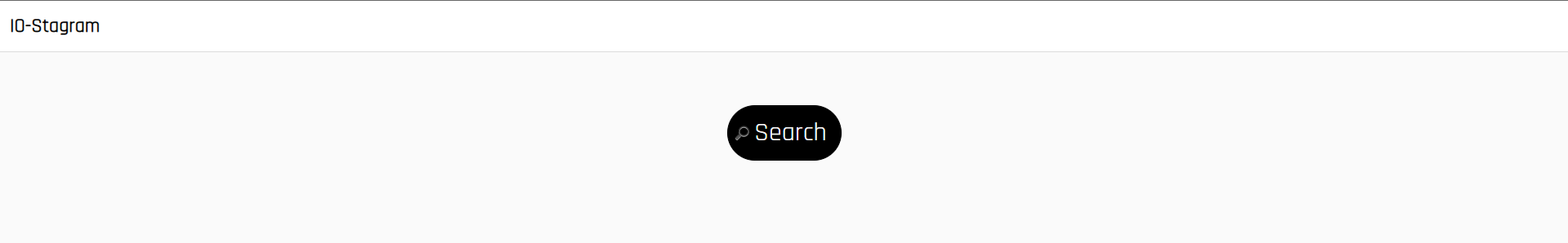
### **5. เริ่มสร้างหน้าแรก หรือ Home**

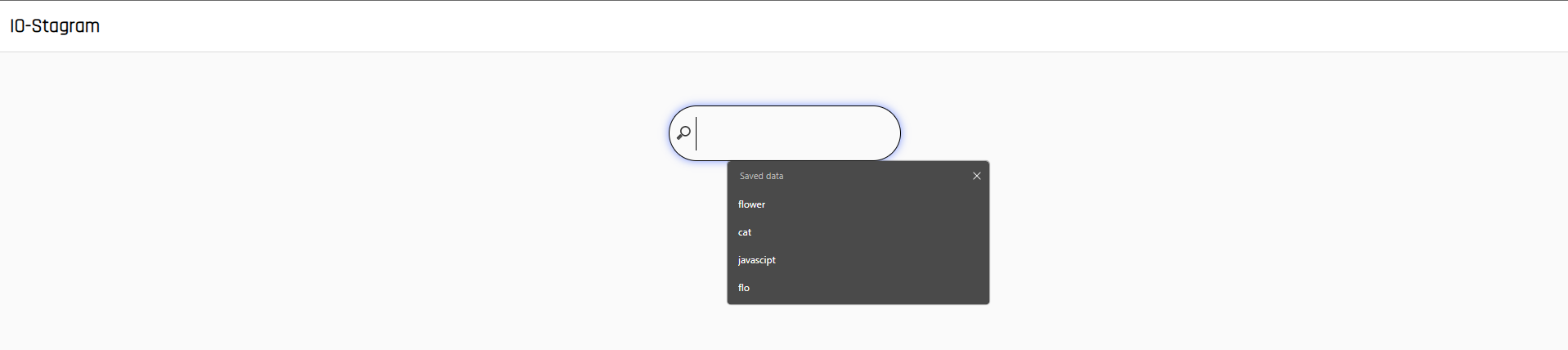
#### 5.1 Home ก็เป็น component อย่างหนึ่ง สร้างเป็น reactFunctionComponent ได้เลย

#### 5.2 มี sub component 1 อันคือ SearchComponent



### **6. SearchComponent**





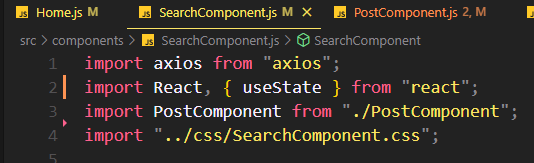


#### 6.1 import สิ่งที่ต้องใช้

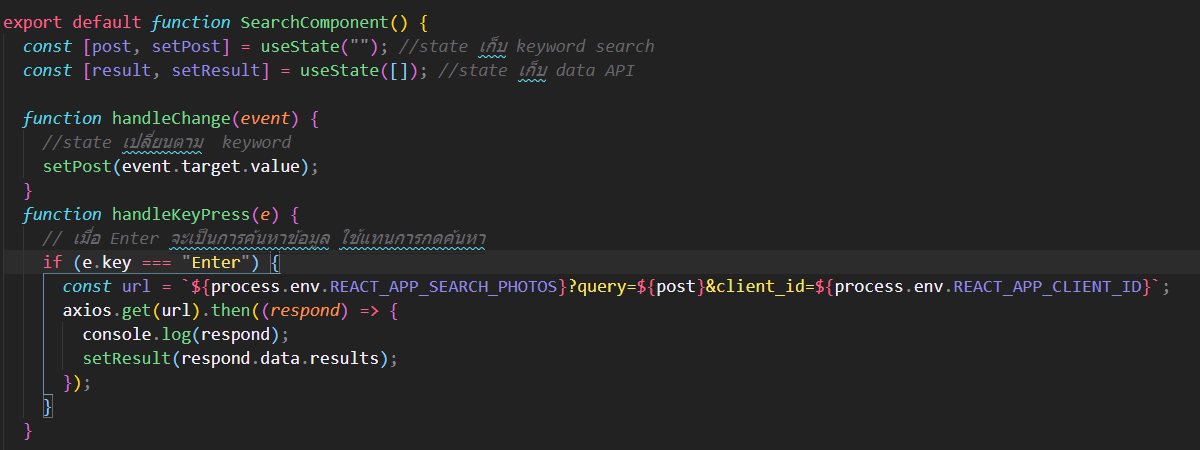
- import package axios เพื่อจัดการกับ API

- import PostComponent เพื่อส่งข้อมูลจากการค้นหาไปให้ในรูป props

- import css เพื่อตกแต่ง



#### 6.2 state และ function



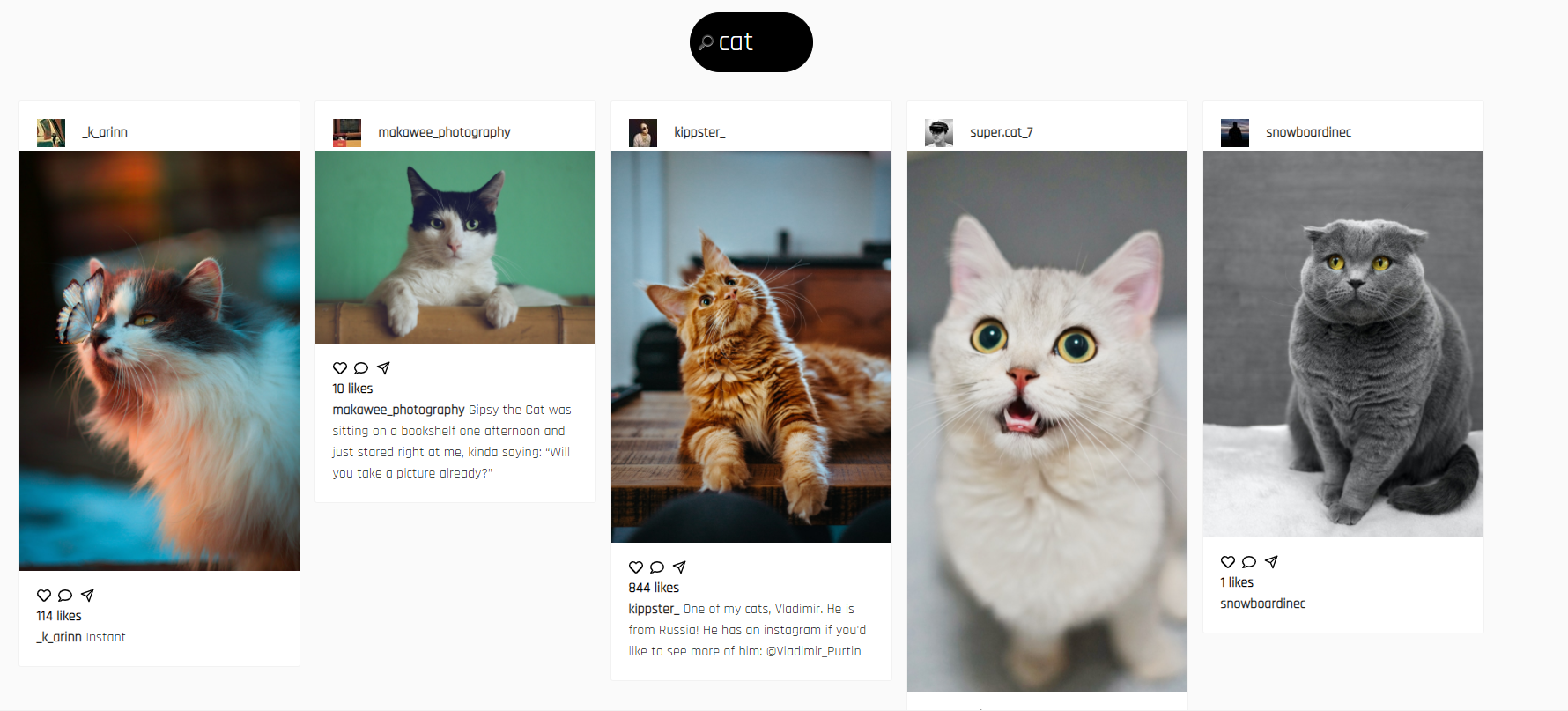
#### 6.3 โครงสร้างช่องค้นหา และ sub component

- แท็ก input มีการเรียกใช้ 2 event คือ onChange , onKeyDown

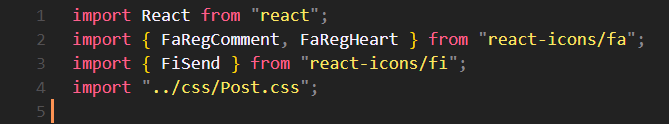
- และส่ง state result ที่เก็บข้อมูลชื่อภาพที่ค้นหาไปให้ PostComponent



### **7. PostComponent**



#### 7.1 import icon และ css สำหรับตกแต่ง



#### 7.2 function การทำงาน

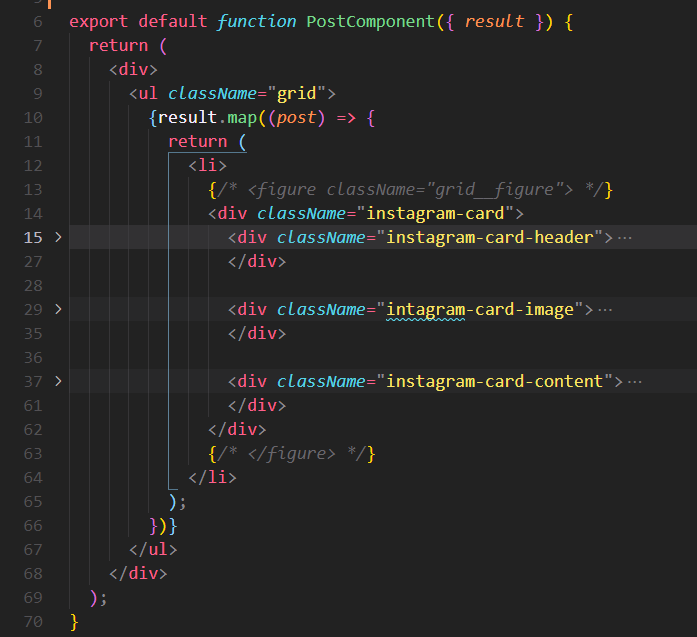
- PostComponent รับ props จาก SearchComponent   
- จากนั้นใช้ function map เพื่อวนลูปหาข้อมูลที่อยู่ใน array เพื่อจะนำข้อมูลออกมาแสดงผล และ call back ออกมาเป็นโครงสร้างของ card ที่ใช้แสดงผล

#### 7.3 องค์ประกอบ Card แสดงผล แบ่งได้ 3 ส่วน

- header สำหรับ แสดงรูปภาพผู้ใช้ และ ชื่อผู้ใช้

- image สำหรับ แสดงรูปภาพที่เกี่ยวข้องกับ keyword ที่ค้นหา

- content สำหรับ แสดงชื่อผู้ใช้ ยอดไลค์ คำบรรยายภาพ

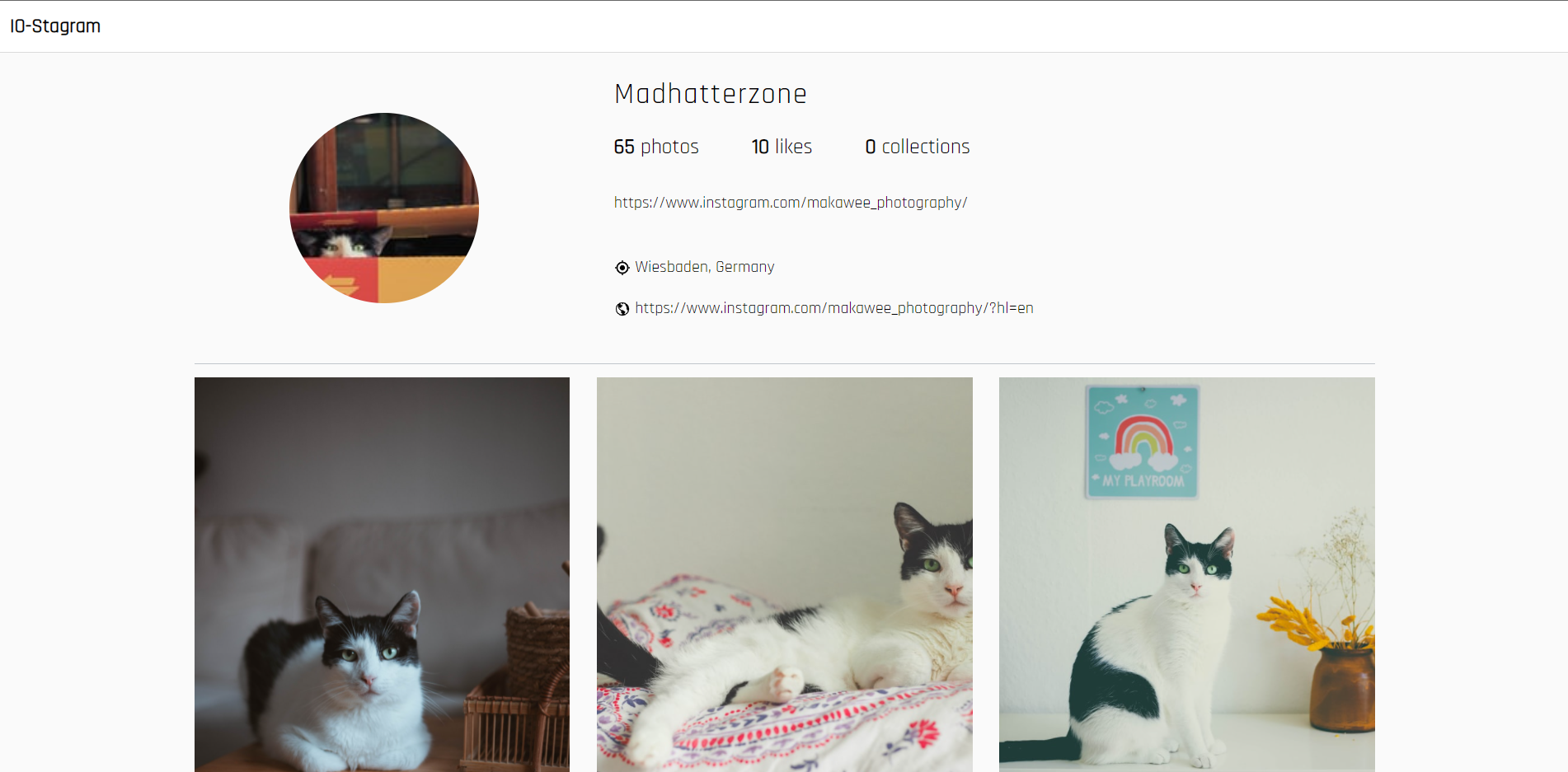


### **8. เริ่มสร้างหน้าโปรไฟล์ผู้ใช้**

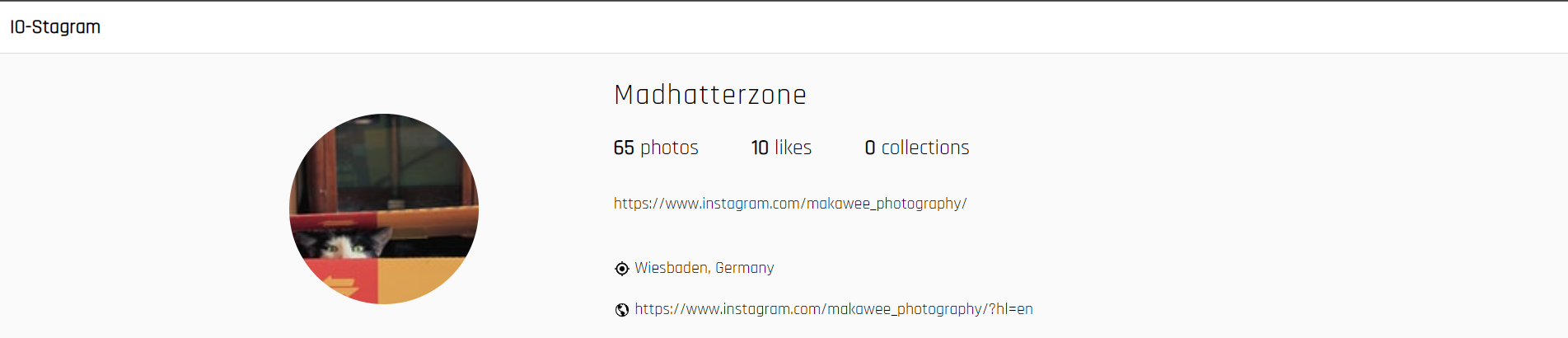
ในหน้านี้แบ่ง component ได้ 2 อัน

- User Profile Component สำหรับแสดงรายละเอียดของเจ้าของภาพ เช่น รูปโปรไฟล์ ชื่อเจ้าของภาพ จำนวนรูปที่มี ยอดไลค์ คอลเลคชั่นที่มี ที่อยู่ ลิงค์โปรไฟล์ที่มีตัวตนจริง เป็นต้น

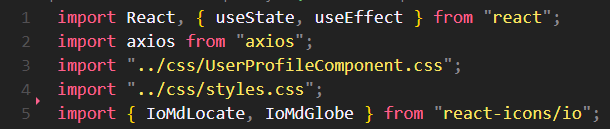
- Photo List Component สำหรับแสดงรูปภาพของเจ้าของภาพคนนี้ว่ามีรูปอะไรบ้าง รูปนี้มียอดไลค์เท่าไหร่



### **9. User Profile Component**



#### 9.1 import package css และ icon ที่จำเป็นต้องใช้



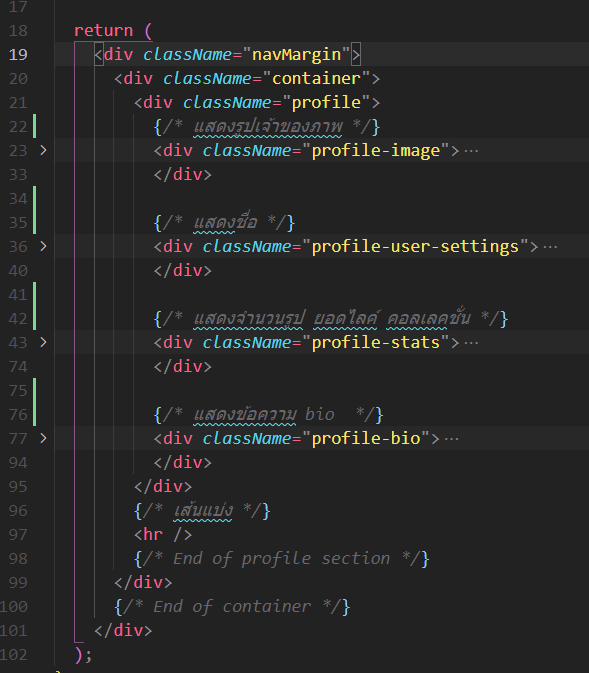
#### 9.2 state และ function



#### 9.3 องค์ประกอบส่วนของ Profile แบ่งเป็น 2 ฝั่ง

- ขวา สำหรับ แสดงรูปโปนไฟล์ของเจ้าของภาพ

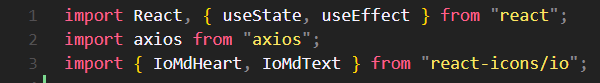
- ซ้าย สำหรับ แสดงข้อมูลของเจ้าของภาพ



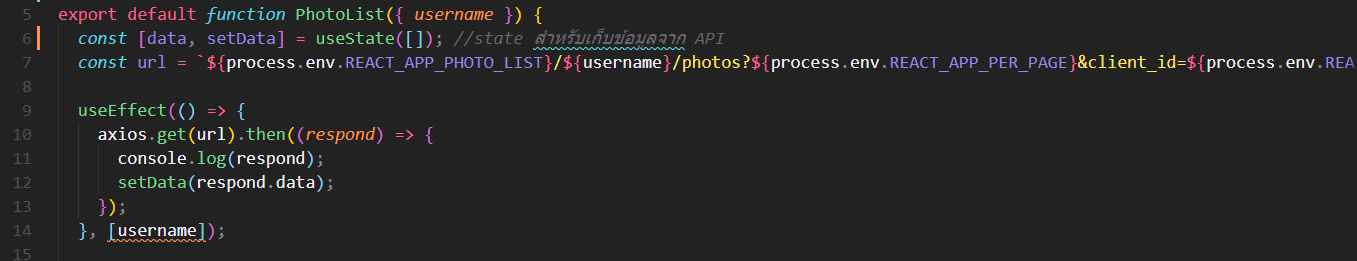
### **10. Photo List Component**



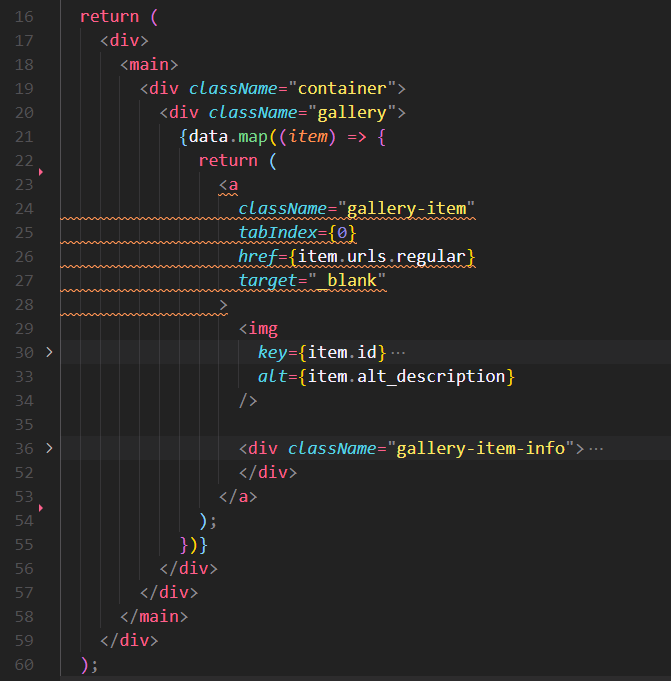
#### 10.1 import package css และ icon ที่จำเป็นต้องใช้



#### 10.2 state และ function

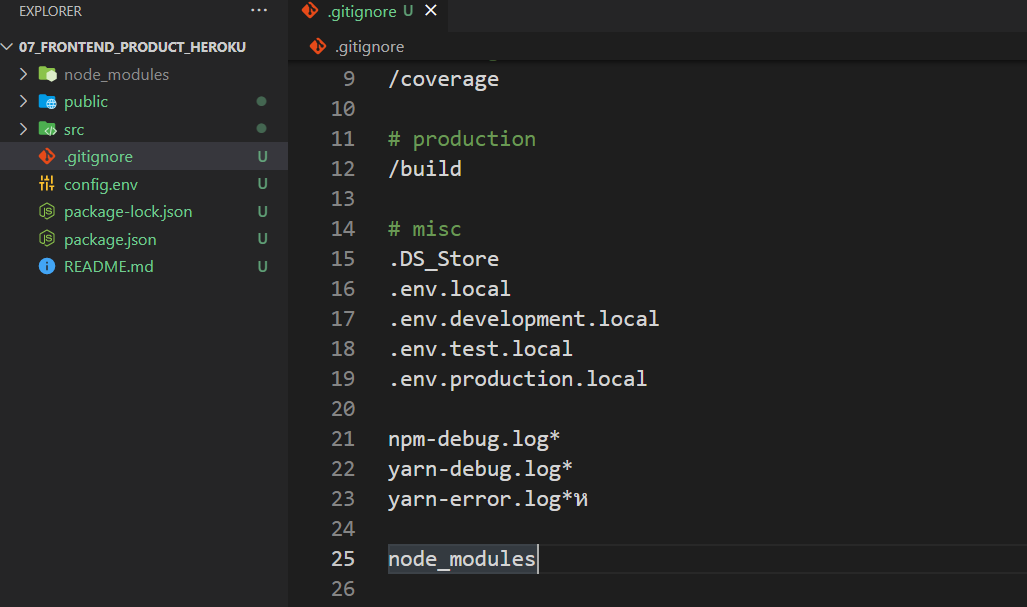


#### 10.3 โครงสร้างของ gallery ที่ใช้แสดง ภาพของเจ้าของภาพ

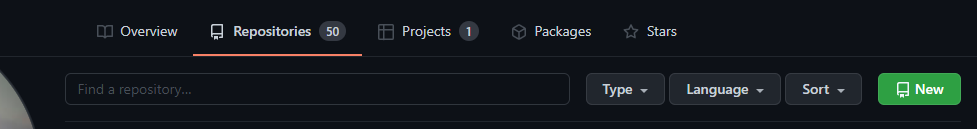


### **11. การ Deploy project ขึ้น server จริง**

#### 11.1 เขียนไฟล์ .gitignore เพื่อไม่ให้ git อัพโฟลเดอร์ node\_modules ขึ้นไป มันจะช้าและไม่มีความจำเป็น



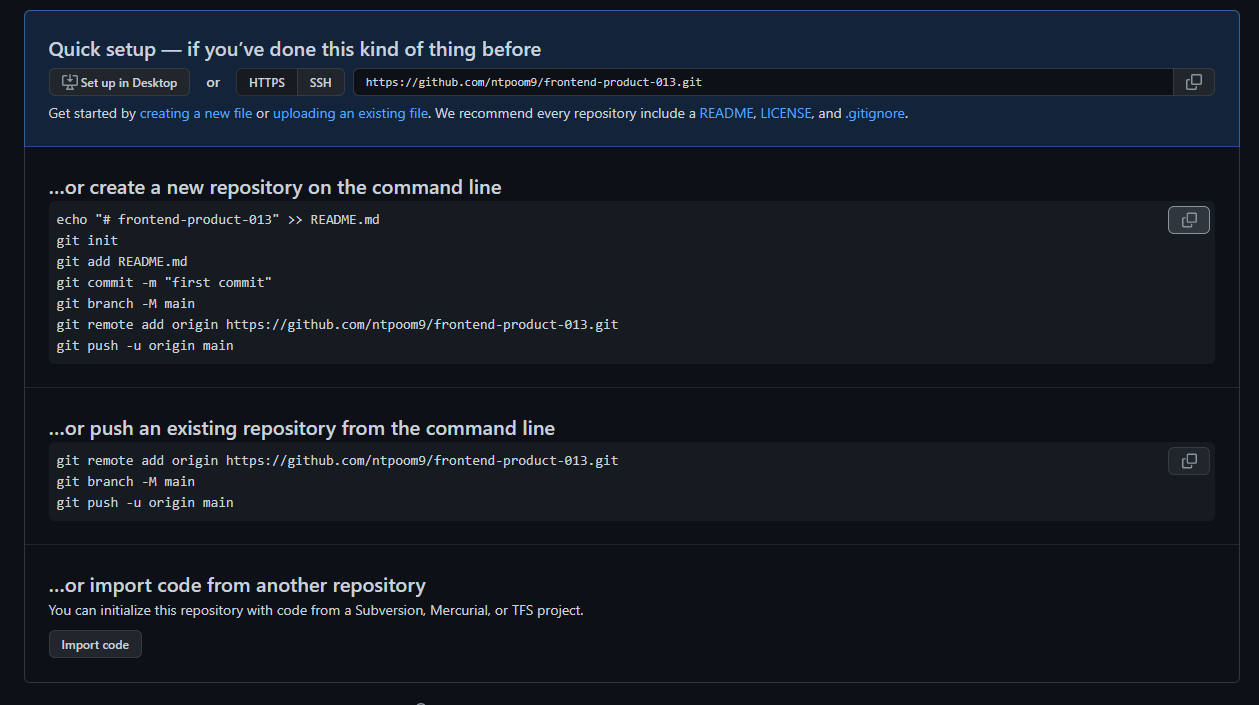
#### 11.2 สร้าง repositories



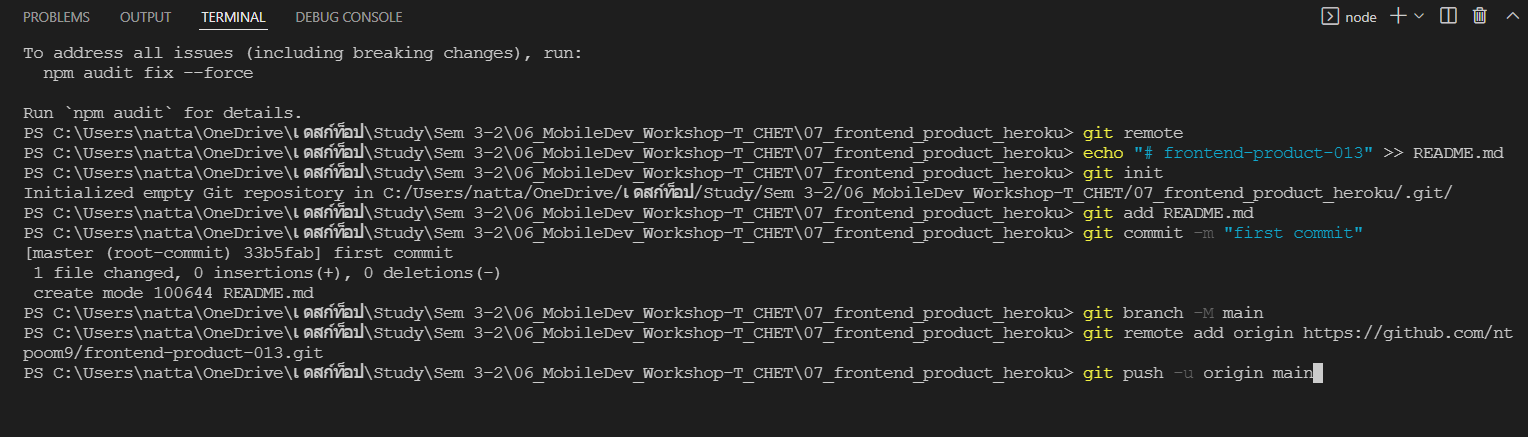
#### 11.3 ตั้งชื่อ repositorise



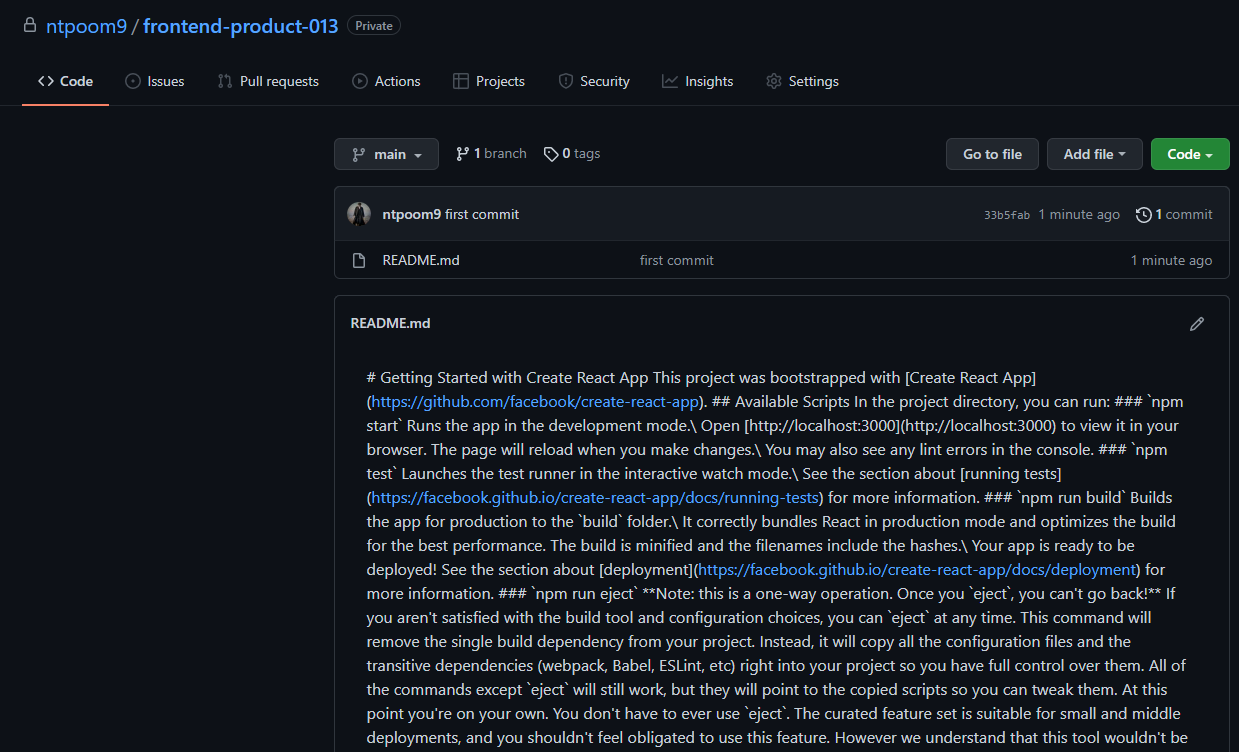
#### 11.4 เชื่อม remote กับ local



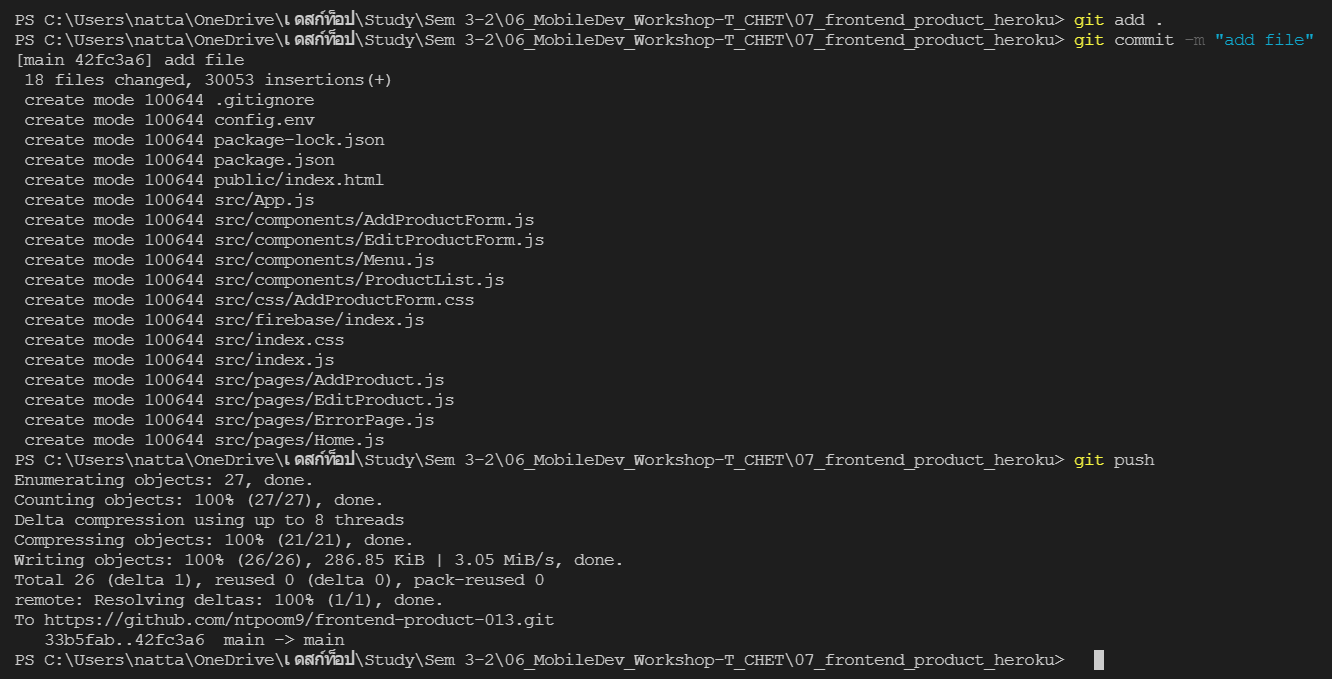
#### 11.5 นำโค้ดส่วนที่ 2 วางใน terminal



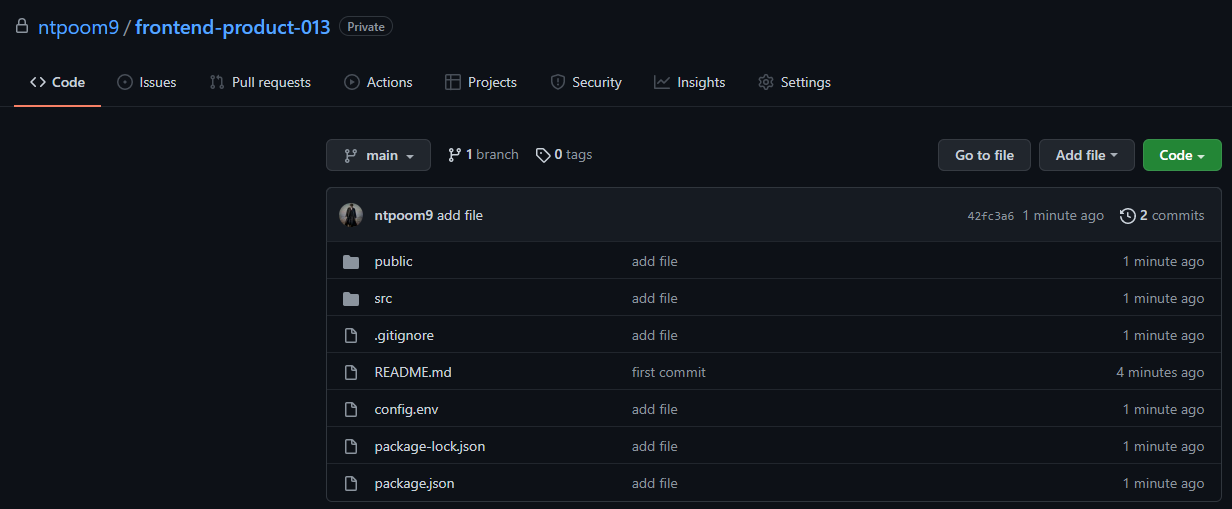
#### 11.6 จะยังไม่มีโค้ด เราต้องมาเพิ่มเองอีกครั้ง



#### 11.7 ใช้คำสั่ง ตามลำดับนี้ เพื่อเพิ่มไฟล์จาก local ขึ้นไปที่ remote(github)

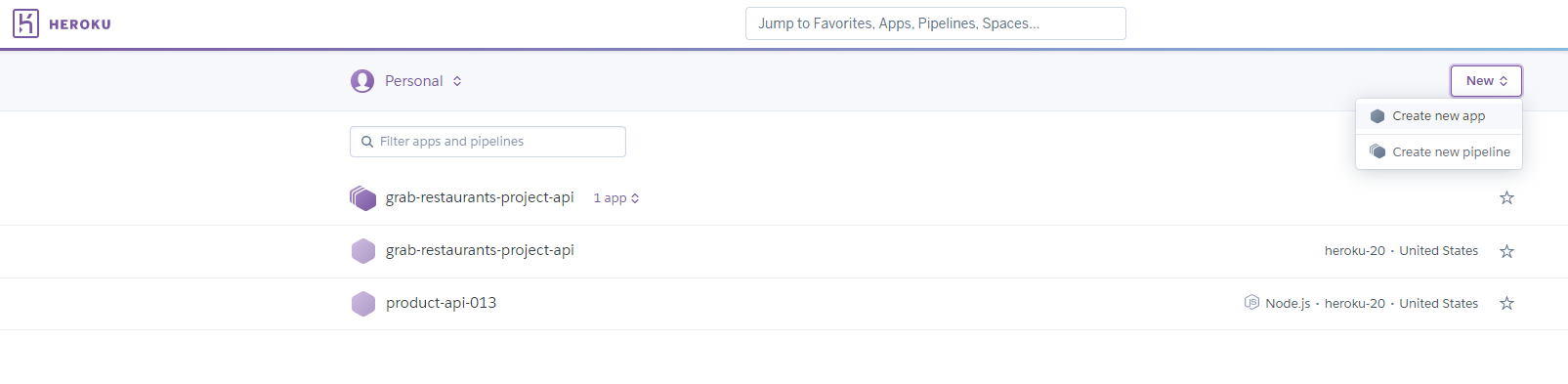


#### 11.8 โค้ดทั้งหมดก็จะถูกเพิ่มมา ยกเว้น โฟลเดอร์ node\_modules



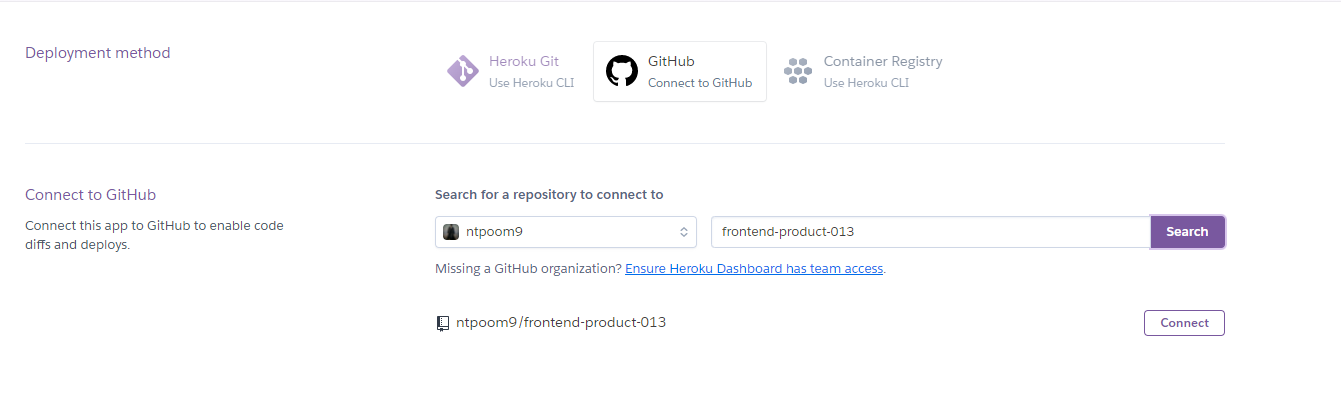
### **12. เชื่อม github กับ Heroku เพื่อเตรียมสำหรับ deploy frontend-unsplash**

#### 12.1 สร้าง app ใหม่ ใน Heroku

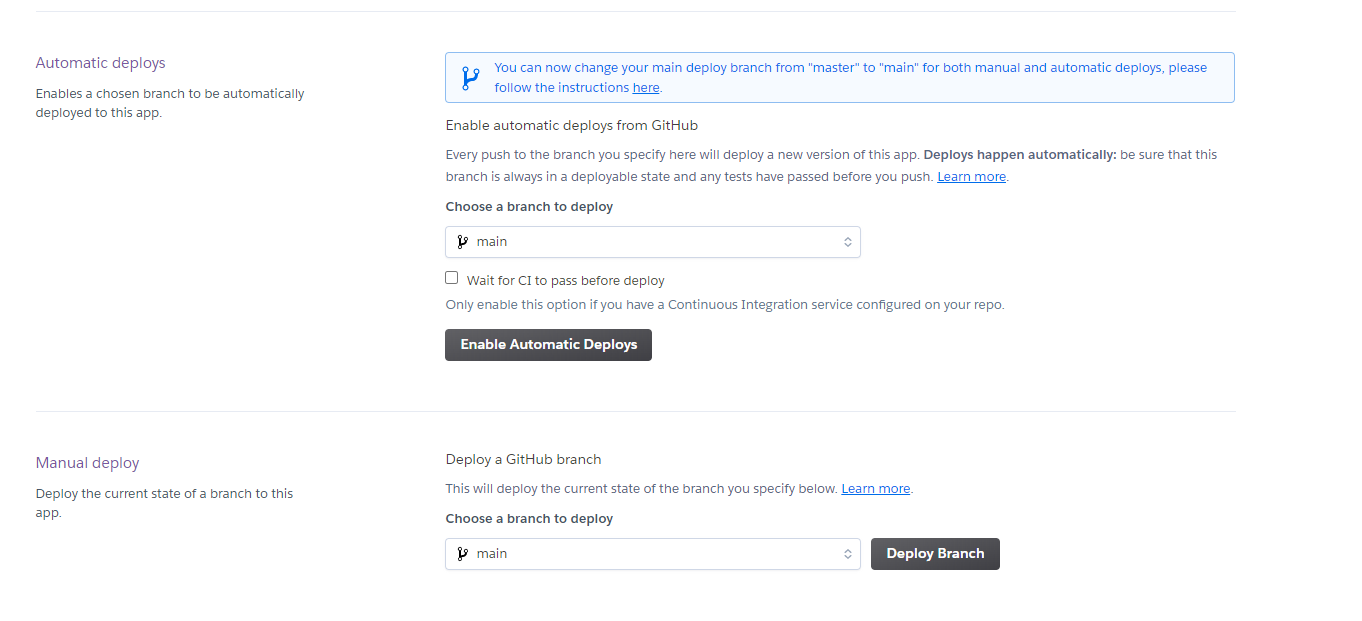


#### 12.2 ตั้งชื่อ app (แนะนำให้ชื่อเดียวกับ repositories ของ github)

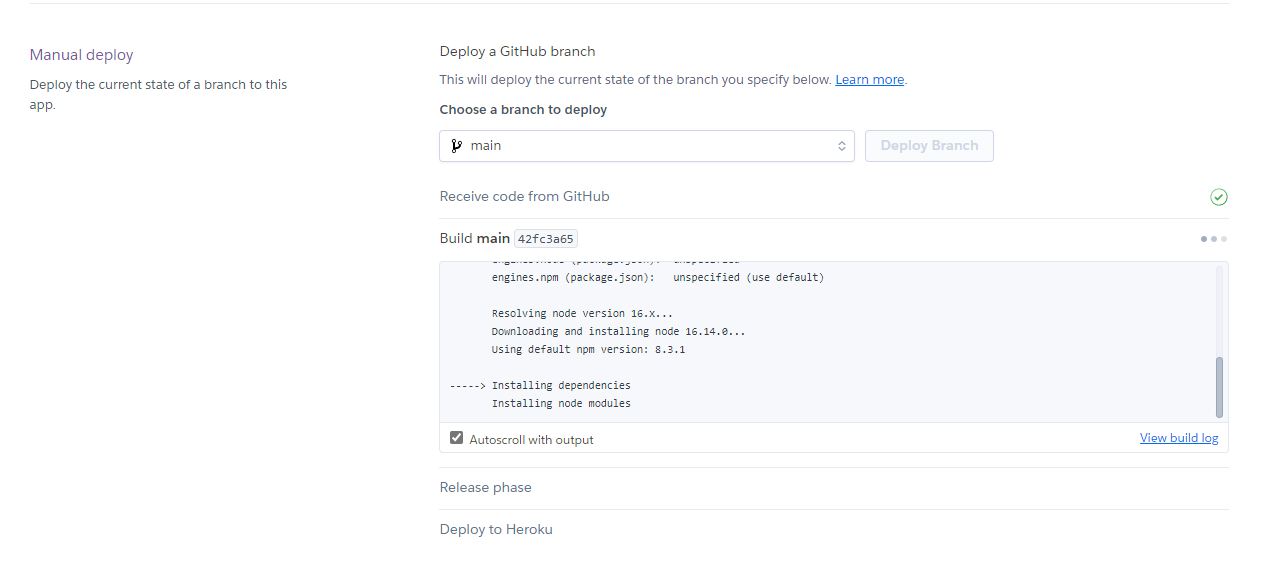
#### 12.3 ทำการตั้งค่าการเชื่อมต่อ Heroku กับ github



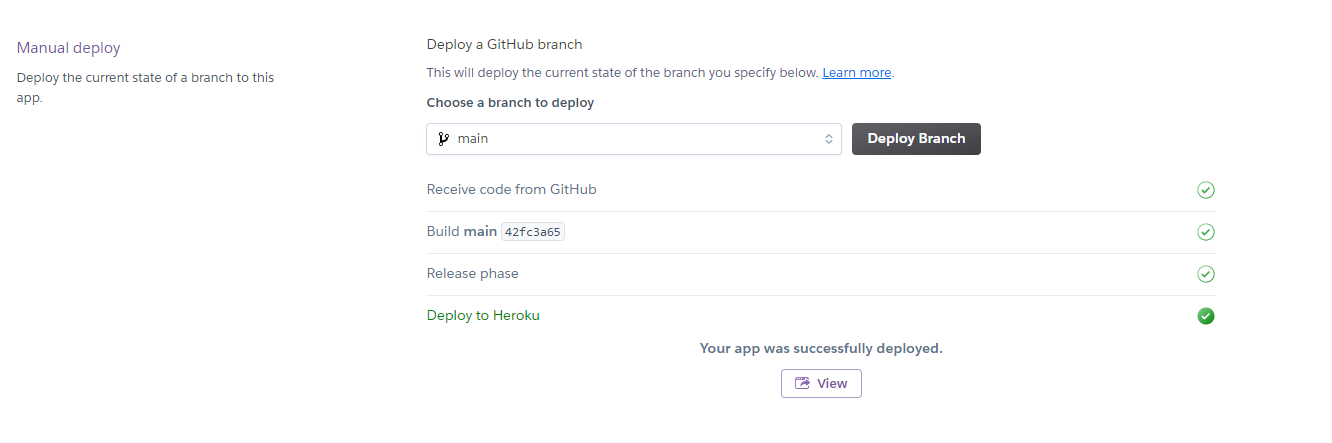
#### 12.4 การ deploy ผ่าน github



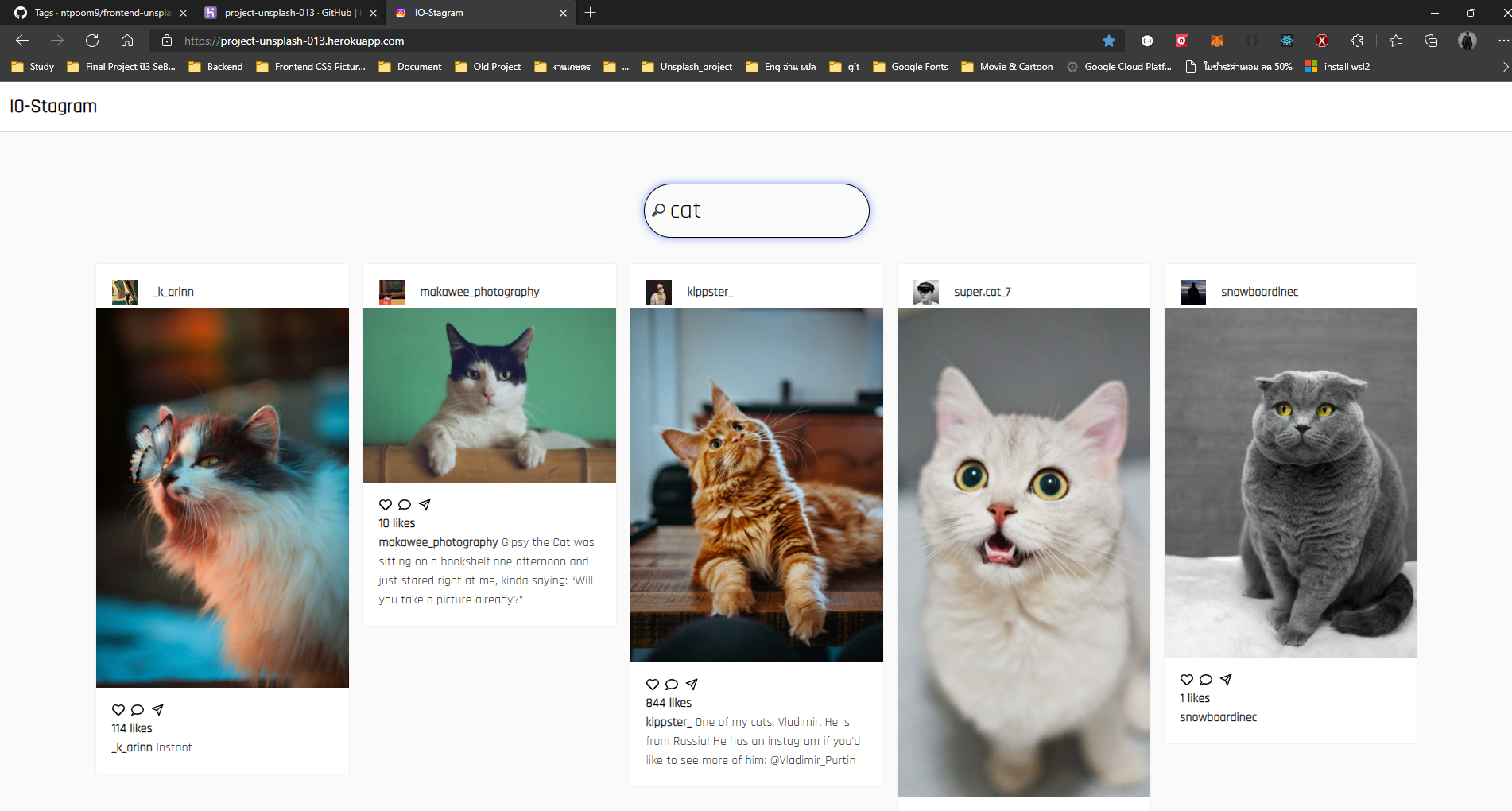
#### 12.5 Heroku จะ build environments ที่จำเป็น สำหรับ app นั้น



#### 12.6 เมื่อ Heroku build เสร็จ กด view เพื่อไปยังหน้าเว็บ



#### 12.7 ผลลัพธ์



### **13. เพิ่ม ตัวแปรที่ใช้ในไฟล์ .env ลงใน Config Vars เพิ่มเติมใน Heroku**

#### 13.1 เข้าไปที่ หมวดหมู่ Setting > Config Vars > Reveal Config Vars

