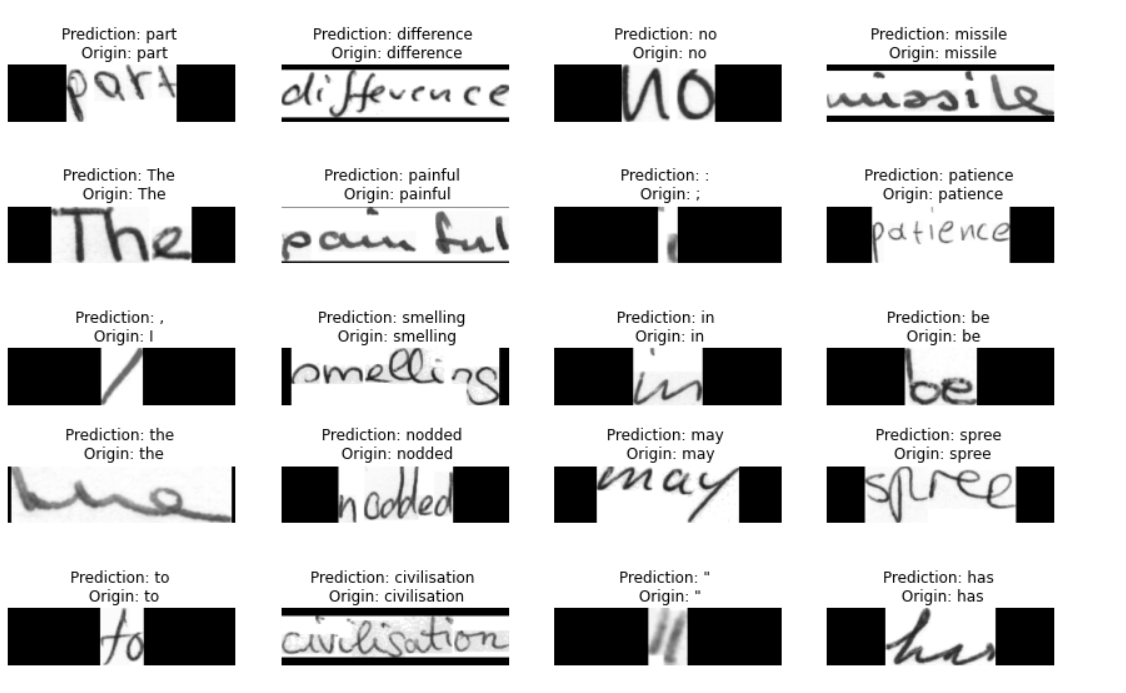
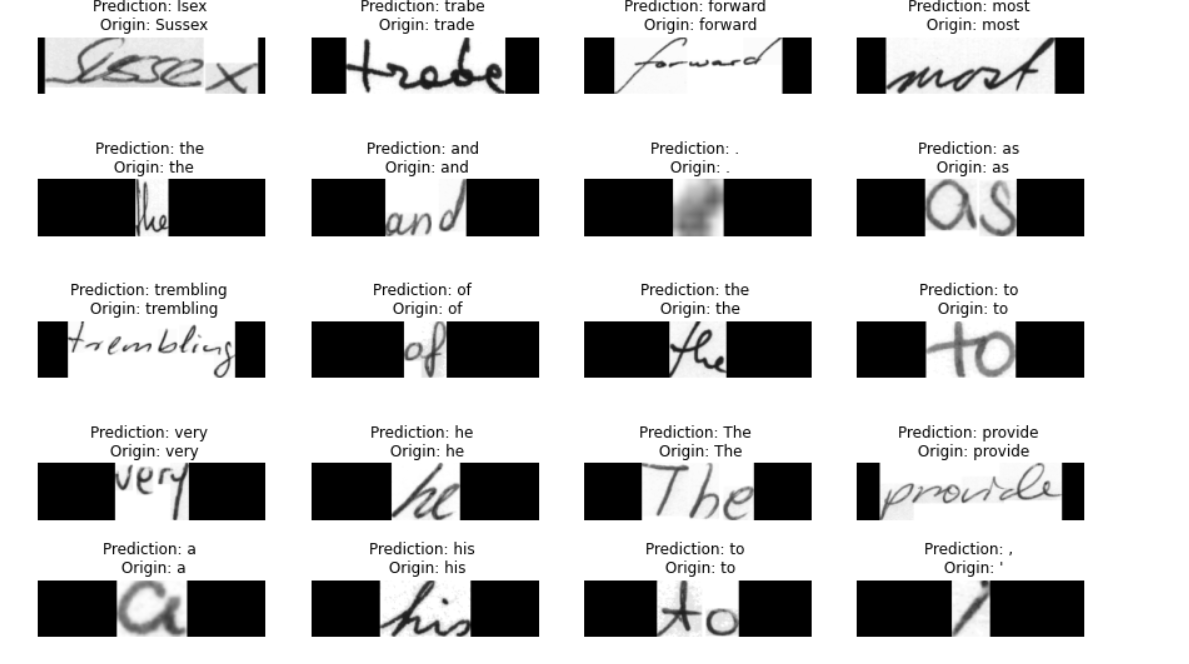
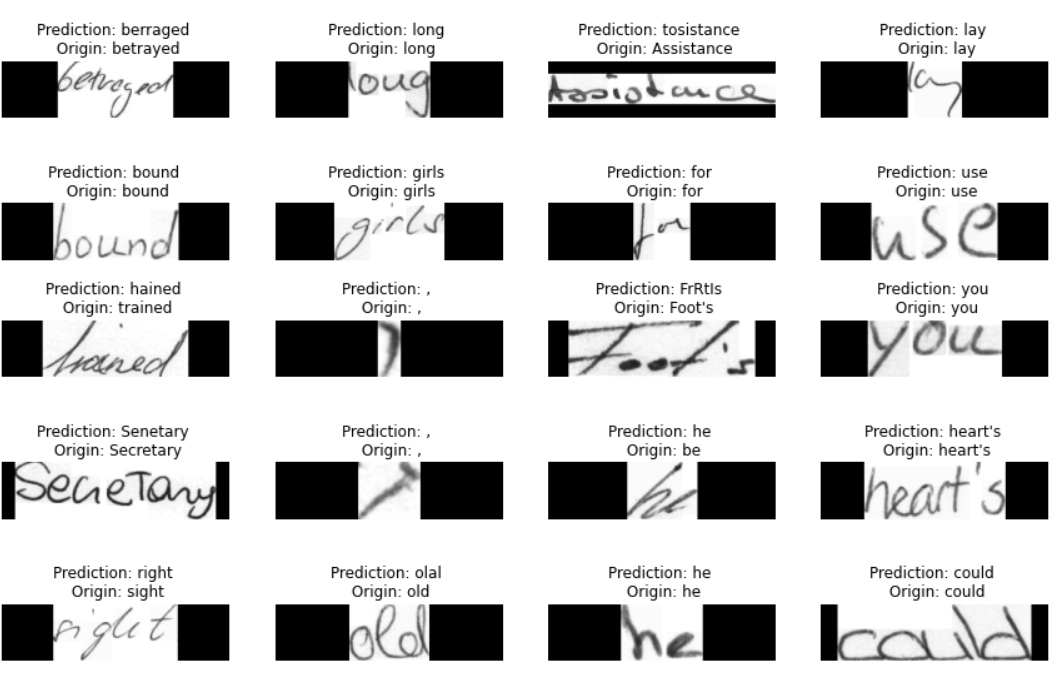
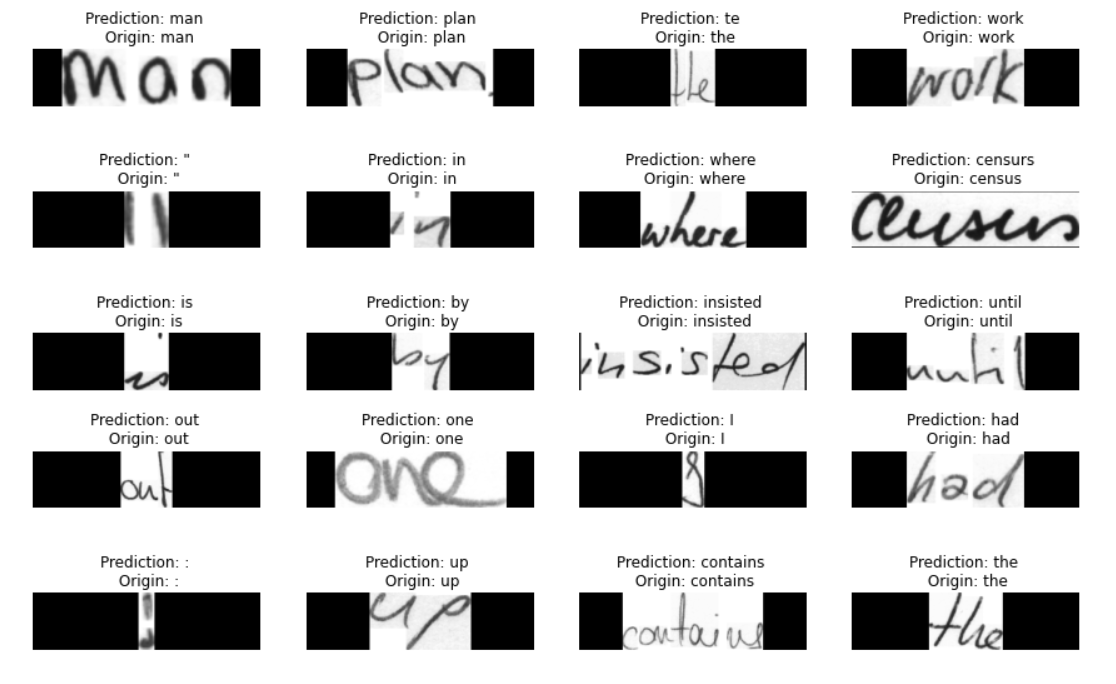
Report Hand Written Recognition

* Một số hình ảnh thu được





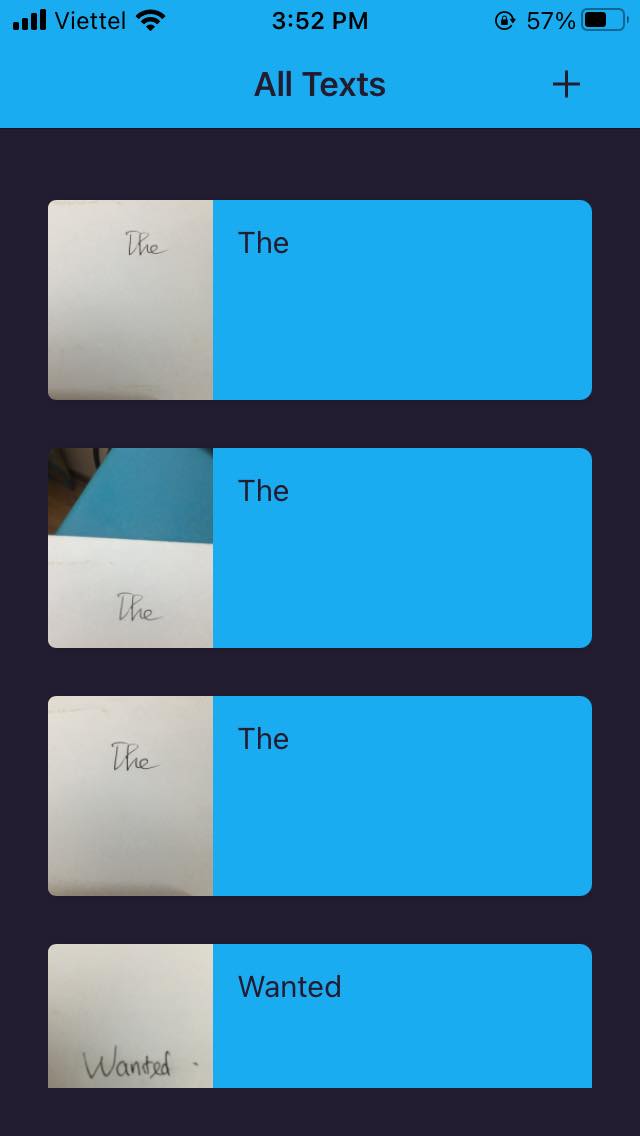




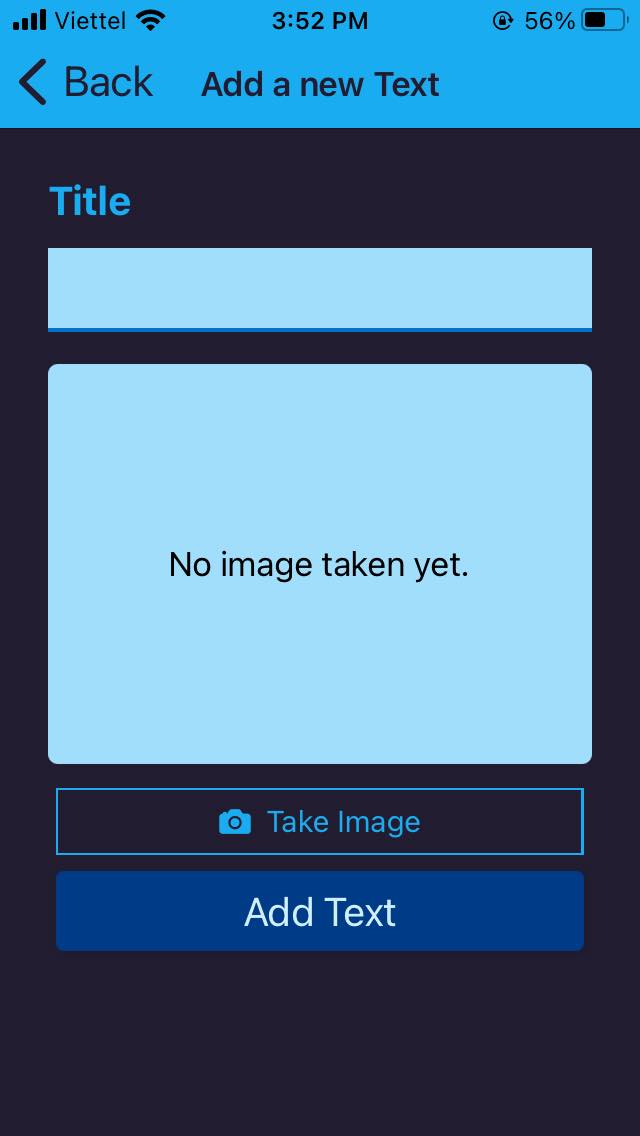
**Application**

1, Giới thiệu

* Technology
* Frontend: React Native (Javascript) => Expo Cli library
* Backend: Flask(Python)
* Code editor: Pycharm, Visual Studio Code
* Environment: Nodejs + Python
* Screen: Ứng dụng của em chia thành 3 screen: MainScreen, AddTextScreen, TextDetailScreen
* Main Screen



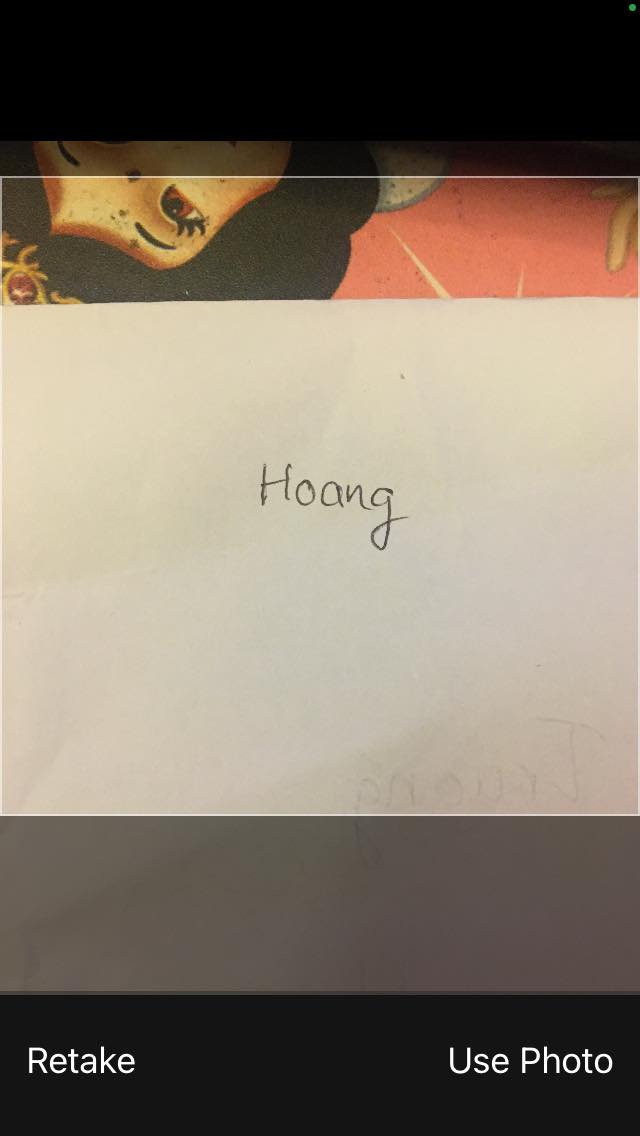
* Main Screen là screen chứa các text và ảnh text do em nhập vào từ AddText Screen.
* Main Screen là màn hình mặc định khi khởi động app
* Từ Main Screen có thể đi qua AddText Screen thông qua nút cộng ở góc phải màn hình, hay đi qua TextDetail Screen thông qua việc chạm vào các component ở màn hình chính
* AddText Screen

****

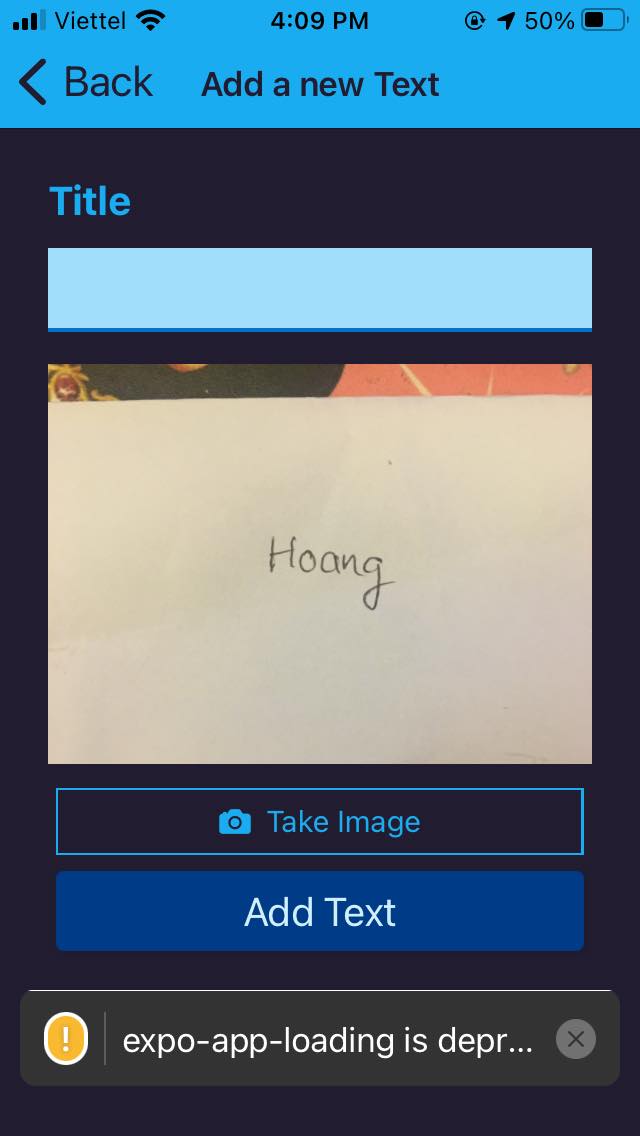
* Chứa form để điền thông tin title và ảnh, trong đó

+ Input Title: Nhập title của ảnh

+ Button Take Image: Khi ấn vào button này thì thiết bị sẽ chuyển sang chế độ camera để chụp ảnh

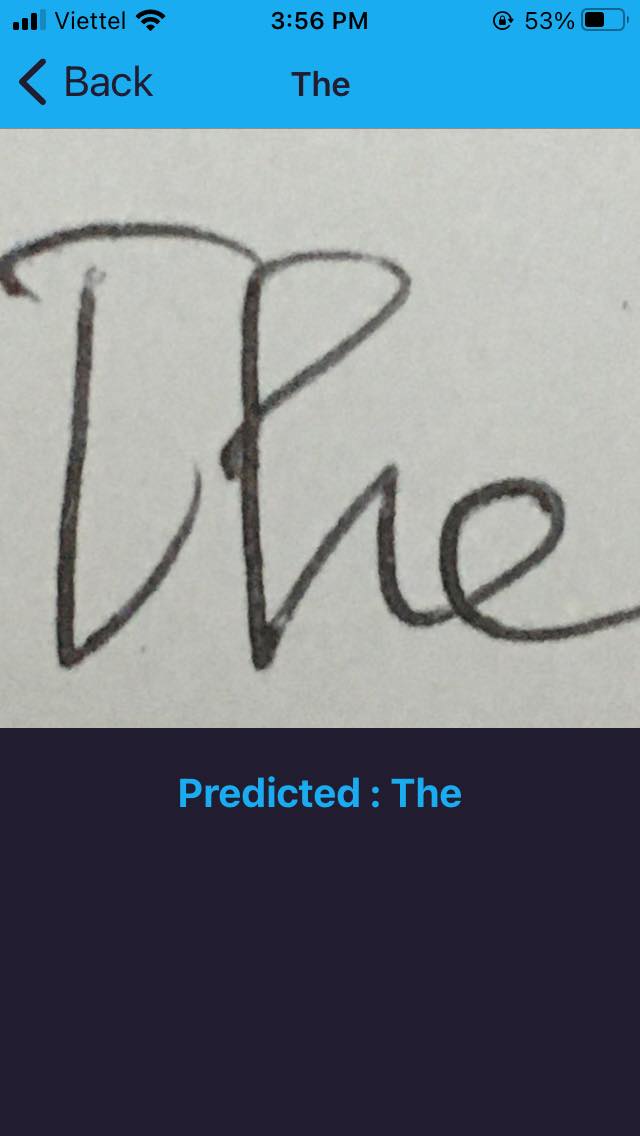
****

+ Sau khi chụp ảnh chúng ta có thể chọn Retake để chụp lại ảnh nếu chưa ưng ý hoặc chọn Use Photo để lấy ảnh (Lưu ý: Do em dùng thư viện ảnh nên sẽ có sự khác biệt giữa android và ios, bên android có thêm chức năng crop ảnh theo tỉ lệ trong khi bên ios thì lại không thể crop ảnh (tỉ lệ 1:1) do đó bên Ios khi xử lý ảnh image\_processiong không tốt dẫn đến predict không cao (em đã thử và chỉ tạo được 1 ảnh đúng) trong khi android do crop được nên xử lý image\_processing khá tốt và predict cho kết quả khá cao). Sau khi click Use photo sẽ trả về AddText Screen với ảnh vừa chụp



+ Button AddText: Submit toàn bộ dữ liệu form vừa nhập lên database và quay lại màn hình MainScreen (hoặc nếu không muốn submit có thể back về MainScreen trên góc trái màn hình)

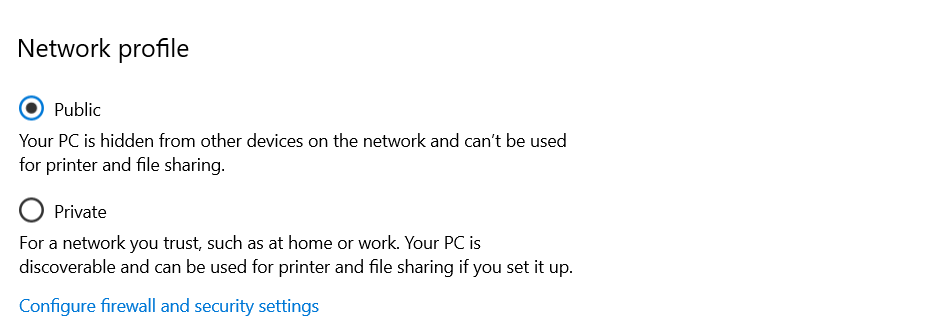
* TextDetail Screen



* Màn hình này sẽ thực hiện nhiệm vụ predict ảnh sang text bằng các gọi API để truyền base64String của ảnh cho backend (Flask) sau đó xử lý base64 string này để tái tạo thành ảnh và preprocessing\_image và cho ảnh đi qua file trọng số đã train để predict ra text và trả về front-end.

2 , Set up

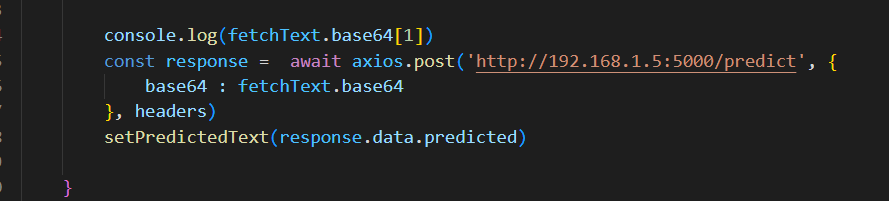
* Requirement: Cài đặt các môi trường và editor cần thiết như Nodejs, npm (đã được tích hợp sẵn trong nodejs), expo-cli , python, pycharm, visual studio code, github
* Sử dụng git để clone project về máy, git gồm 3 branch:
* client: chứa code của react-native (front-end).
* server: chứa code của python (back-end)
* notebook: chứa notebook của research
* Frontend:
* Bên trong folder gr1 (branch client), mở terminal gõ npm install để cài đặt package cho front-end. Danh sách package và version tương ứng đã được tích hợp trong file package.json
* Sau khi cài đặt xong thư viện thì gõ lệnh npm start để chạy ứng dụng front-end.
* Front-end có thể giả lập trên máy tính (AndroidStudio cho Window và XCode bên MacOs (không dùng được do XCode không hỗ trợ chạy Camera trên thiết bị giả lập của MacOs)) hoặc có thể chạy trên thiết bị thật thông qua App Expo (Terminal sẽ gửi về 1 mã QR để scan truy cập vào trong app)
* Backend: Mở branch server bằng Pycharm và cài đặt toàn bộ thư viện cần thiết. Danh sách thư viện có trong file requirement.txt.
* Kết nối frontend và backend:
* Đầu tiên phải đảm bảo cả frontend và backend cùng kết nối đến 1 mạng LAN, sau đó vào wifi trên máy tính chọn chế độ properties của wifi đang kết nối và chuyển mode sang public nếu đang ở private (rất quan trọng vì nếu để ở private thì server sẽ không nhận request bên client để xử lý)



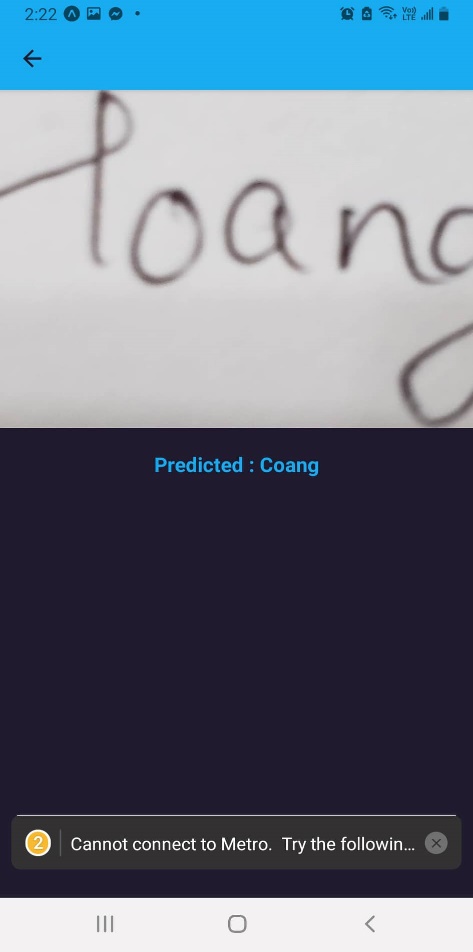
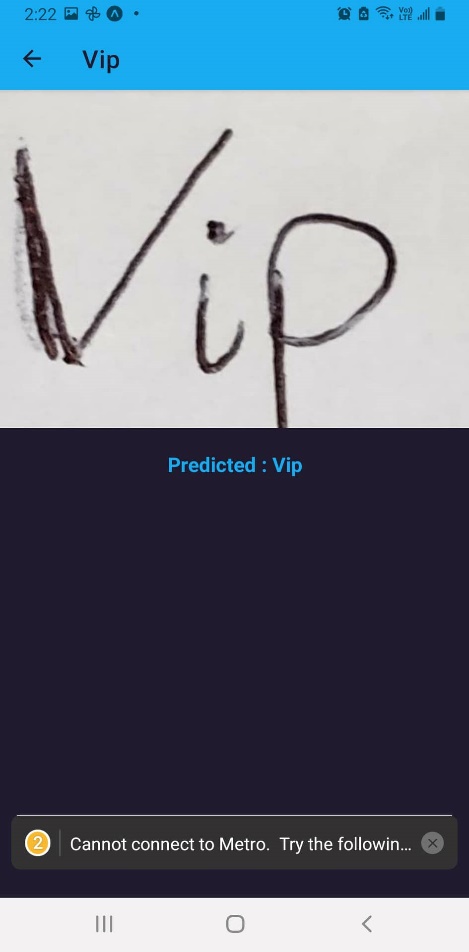
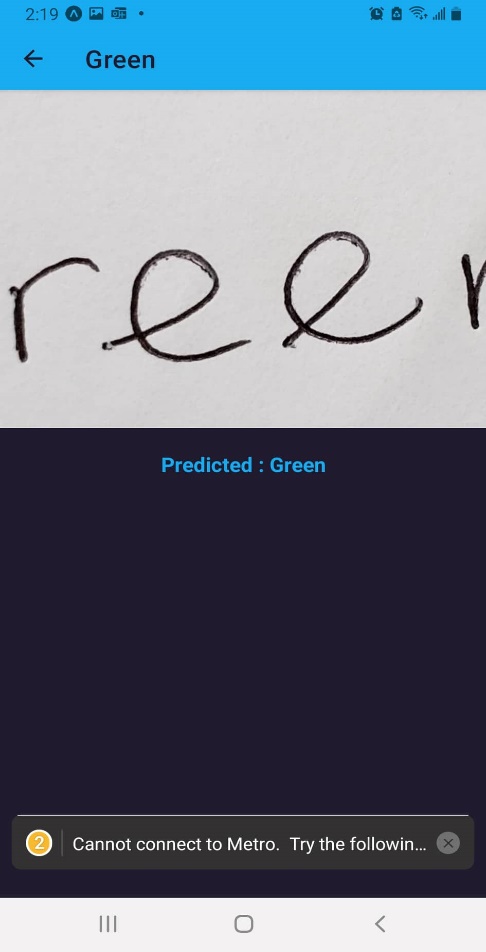
* Cũng trong properties này di chuyển đến IPv4 address 
* Bên backend thì ở file main.py thay đổi host trong app.run() thành IP

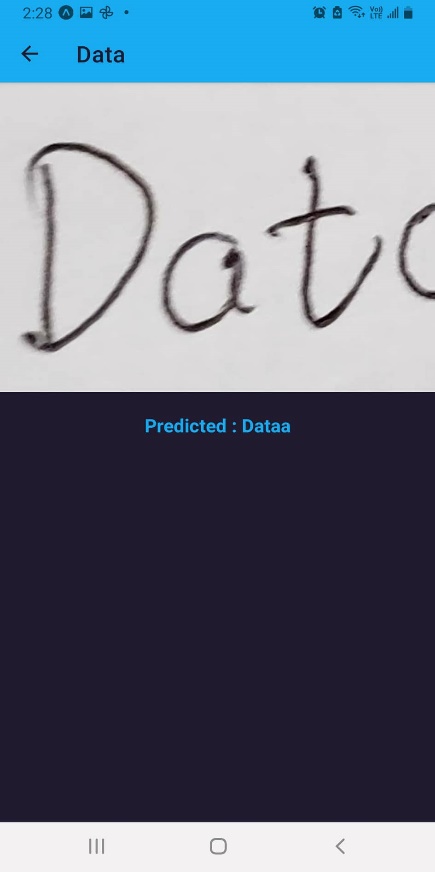
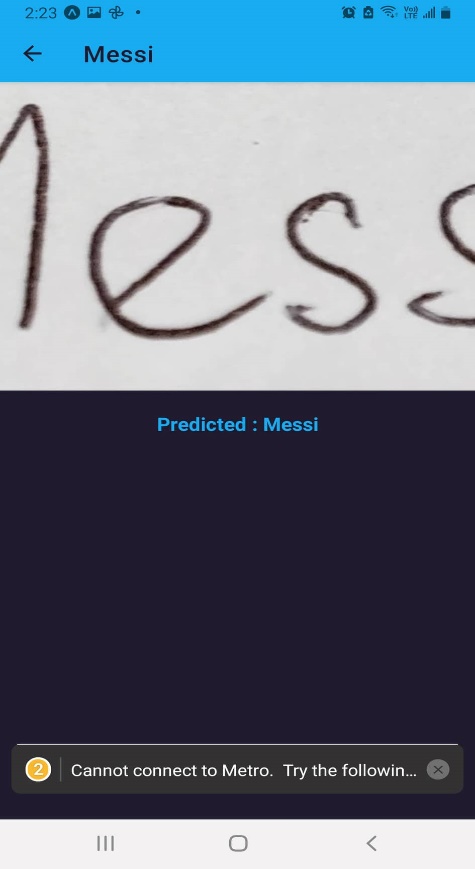
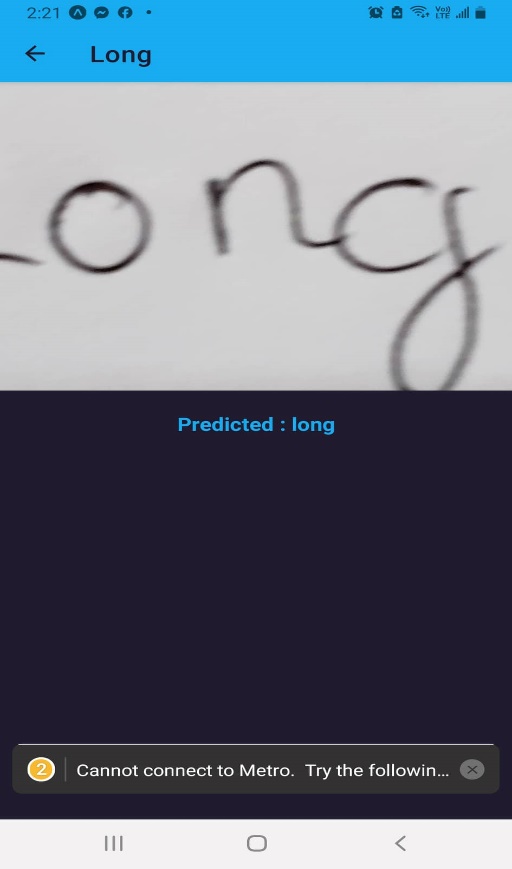


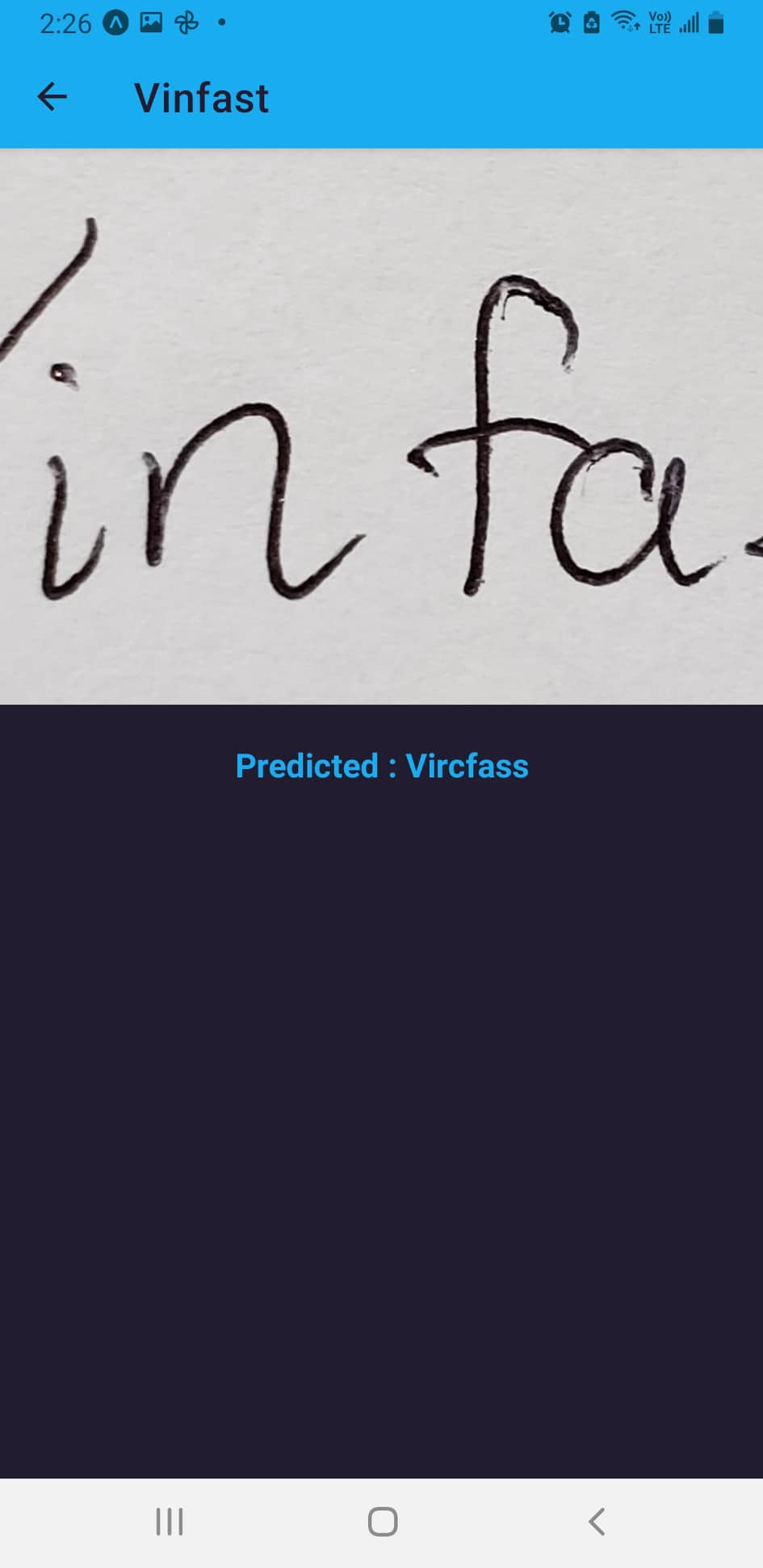
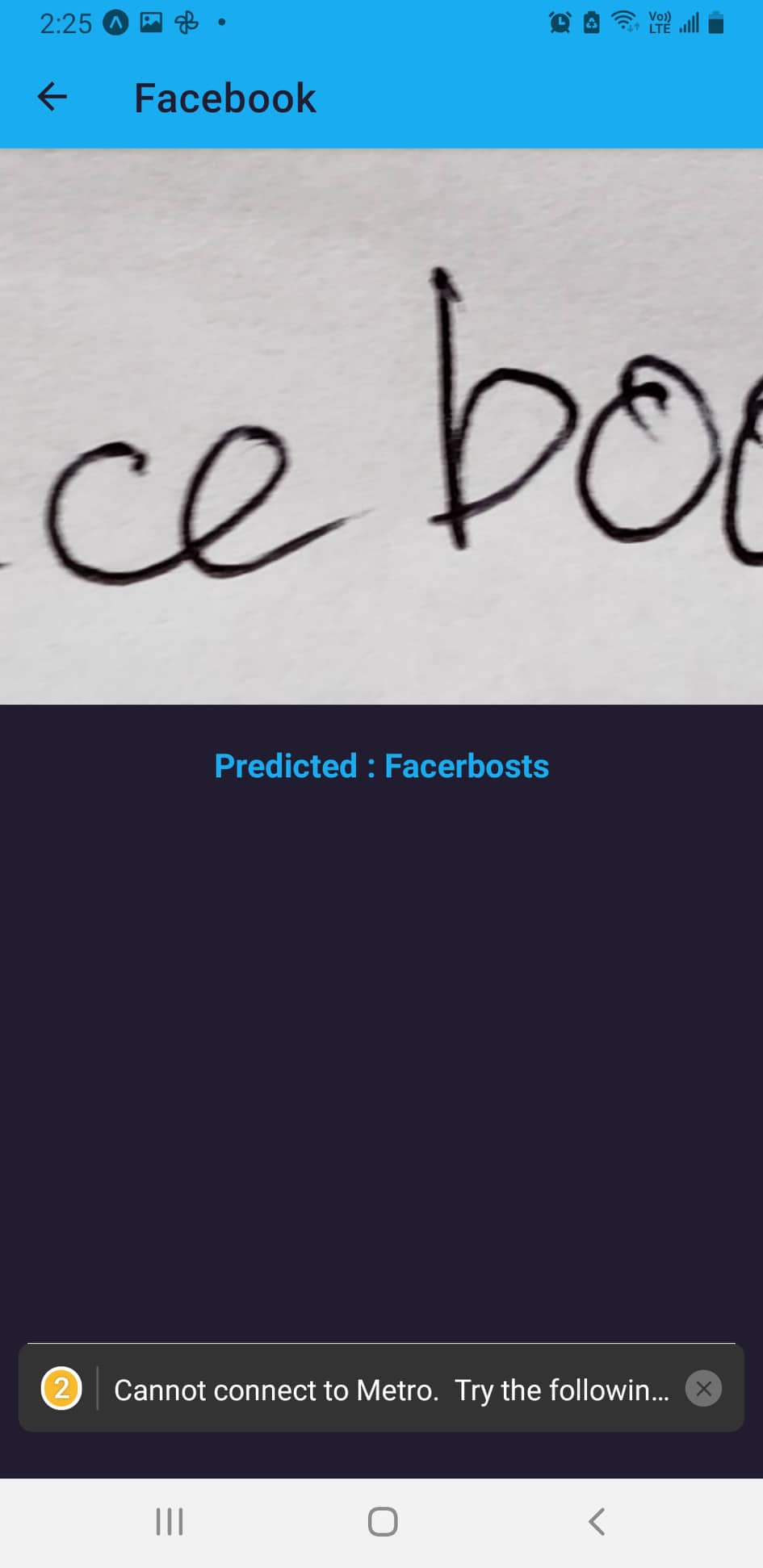
* Bên frontend thì ở file TextDetail.js thay đổi post API sang IP



3, Demo







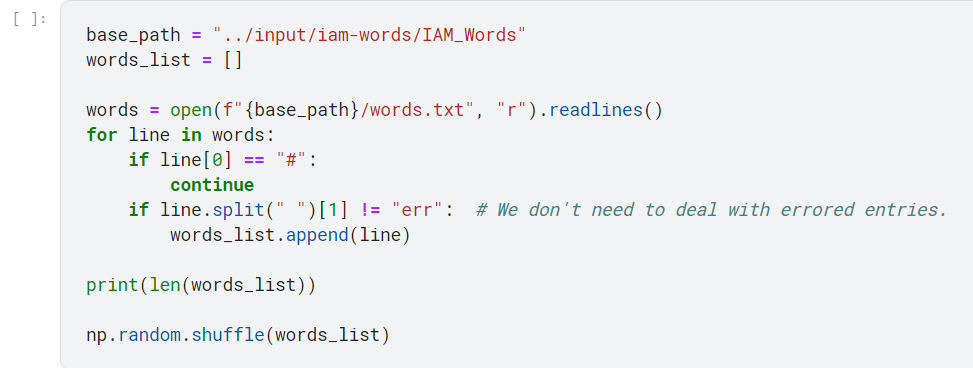
Link github: https://github.com/HoangPH10/gr1.git

**VI, Phụ lục**

**Import library**



**Đọc tất cả text trong file txt bao gồm link ảnh và label**

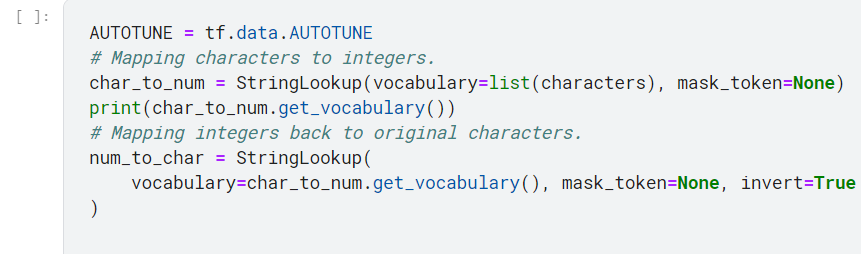


**Xử lí dữ liệu text để lấy link ảnh và label**



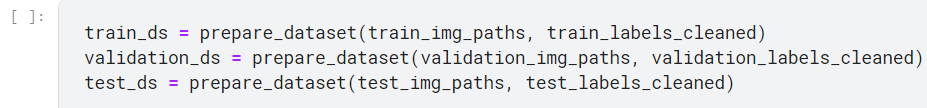
**Lấy dữ liệu characters trong file txt và chuyển đổi thành token**



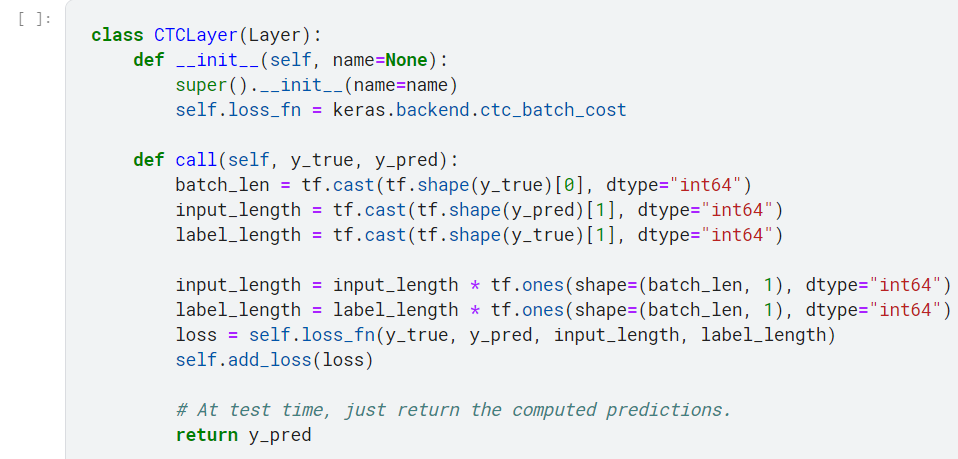


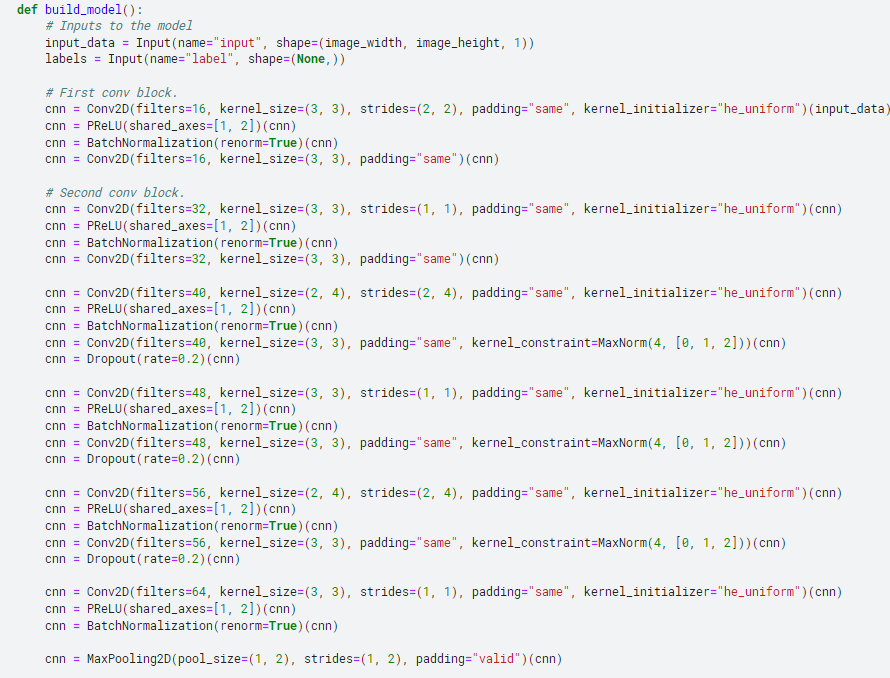
**Xử lý ảnh đầu vào bao gồm grayscale, resize, padding processing, chuyển label thành token**





**Xây dựng CTC layer và build model**







**Compile and fit model**



**Save weight model**



**Predict**

