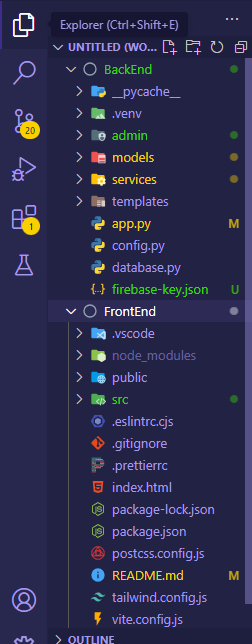
# BÁO CÁO BÀI TẬP LỚN

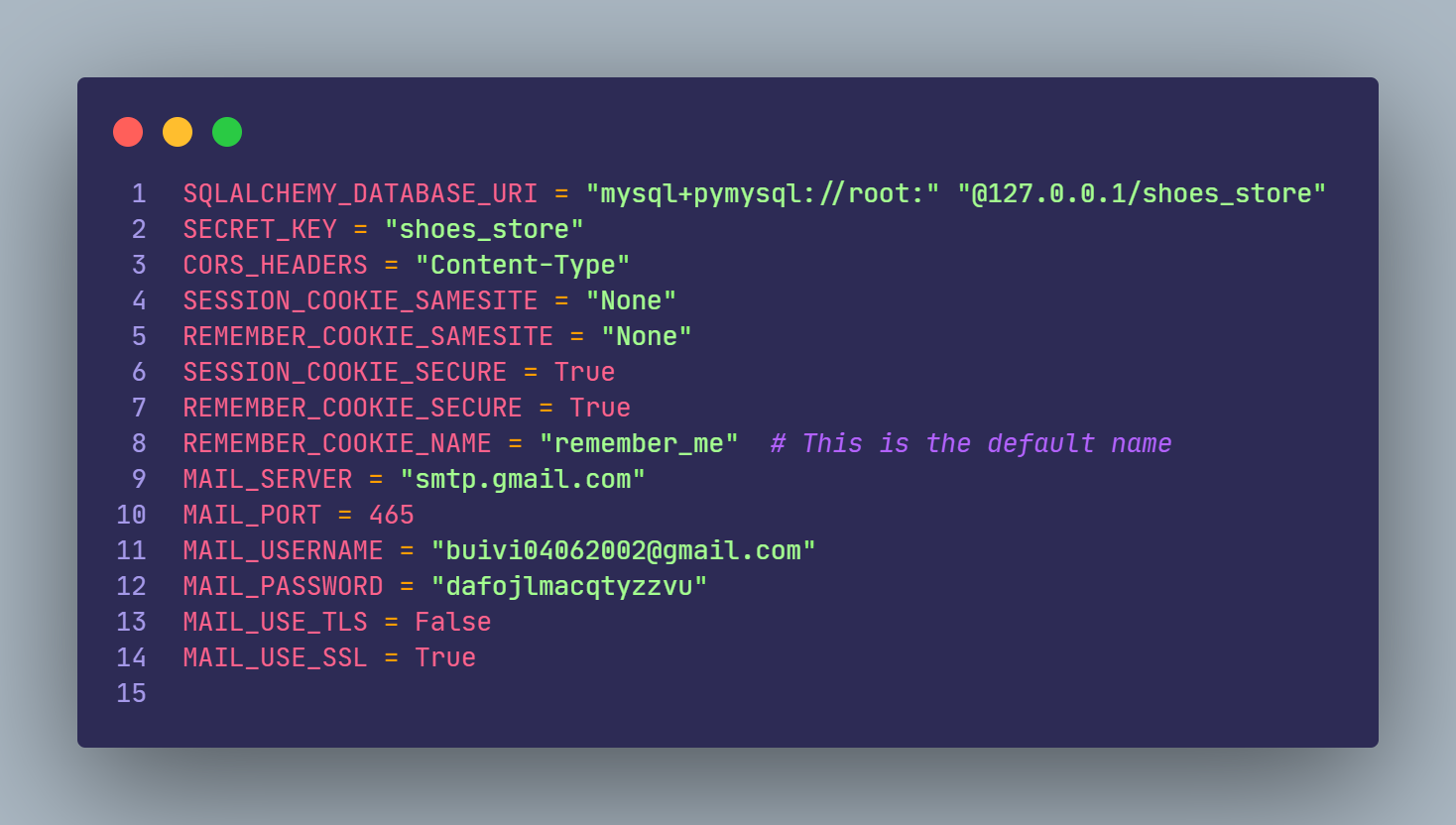
Đề bài : Xây dựng ứng dụng Python Web bao gồm một website bán hàng CRUD Login/Logout ShoppingCart Paging Search Rest API. Ứng dụng bao gồm Python Web – Bootstrap – MySQL. Sinh viên có thể chọn làm ứng dụng Python Web FullStack (sử dụng Django|Flask|Pyramid), hoặc xây dựng ứng dụng với Frontend là ReactJS/Angular/VueJS, còn Backend có thể sử dụng Python Django|Flask|Pyramid. Viết tài liệu mô tả chi tiết các bước xây dựng ứng dụng step-by-step

# KHỞI TẠO DỰ ÁN



Dự án này được viết bằng flask (tạo api) và giao diện được xây dựng bằng React JS

# KẾT NỐI CƠ SỞ DỮ LIỆU



Trong folder Backend ta tạo file config.py chứa các cấu hình của app

Đầu tiên ta cài flask\_sqlalchemy và pymysql: pip install flask-sqlalchemy pymysql

Để kết nối đến xampp mysql ta sử dụng thêm thư viện pymysql để kết nối đến xampp ta khai báo biến SQLALCHEMY\_DATABASE\_URI như trên

Các biến config ở dưới lần lượt là các cầu hình cần thiết cho flask-cors, flask-login và flask-mail

Về trang quản trị ở đây ta sử dụng flask-admin

Tạo model User



Tiếp theo ta tạo file templates/admin/index.html



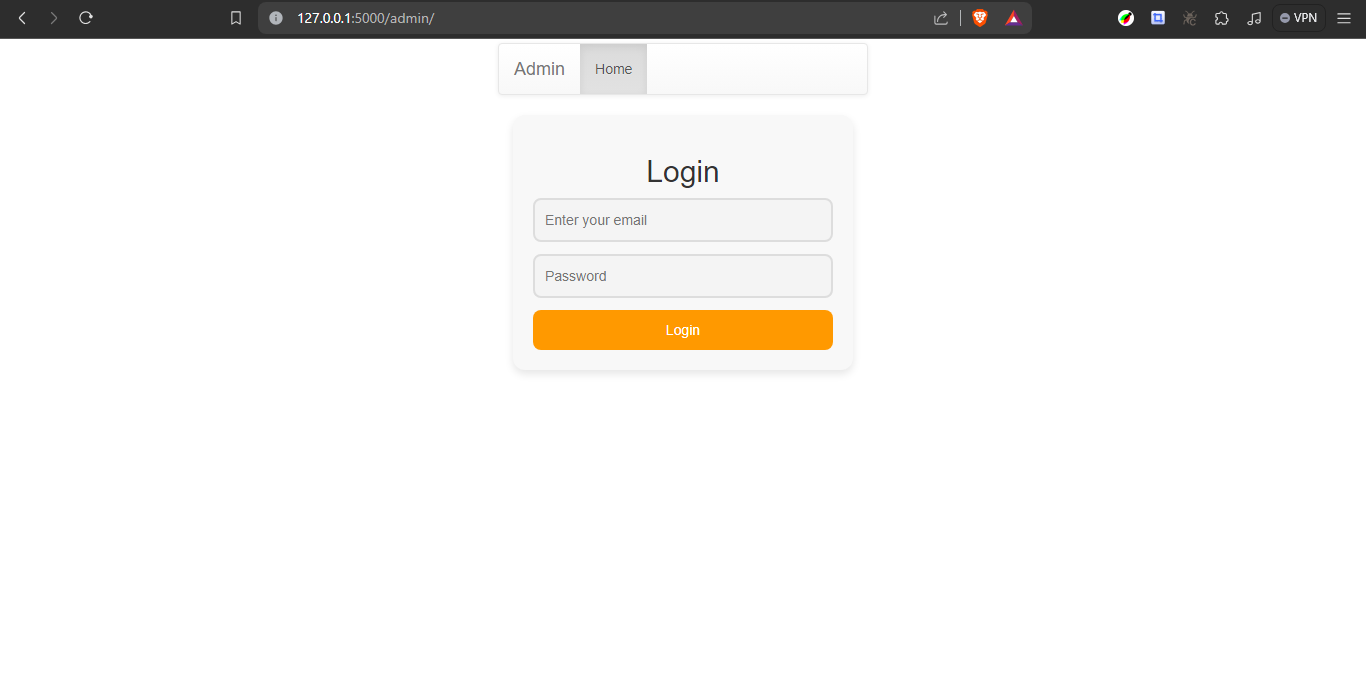
Và file templates/admin/login.html để tạo form login cho người dùng muốn truy cập vào trang quản trị



Tiếp theo ta tạo url cho việc đăng nhập admin để kiểm tra xem email và password và role của người dùng có hợp lệ không.



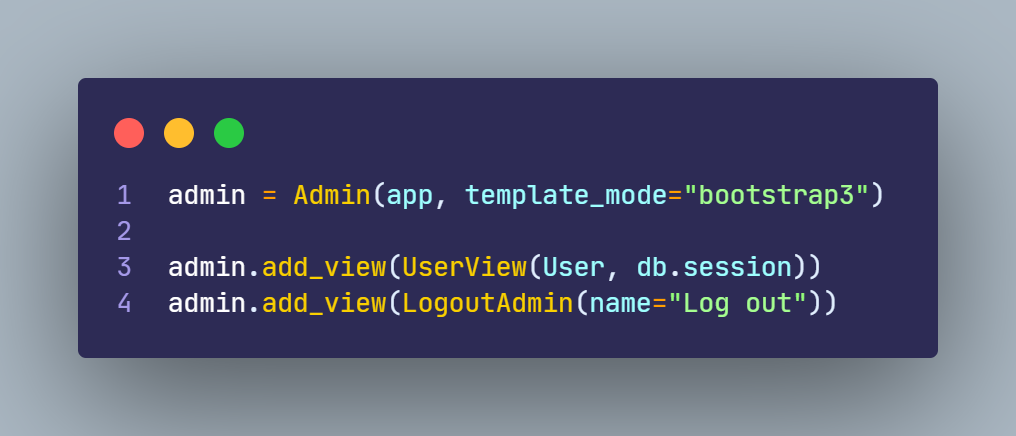
Giao diện tại trang quản trị



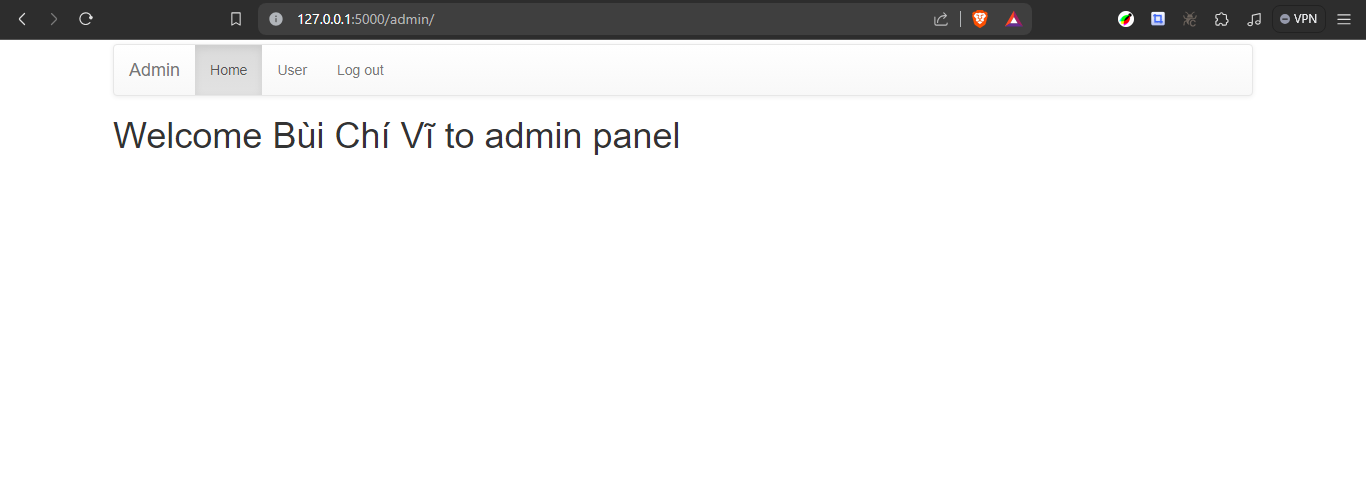
Tiếp theo tạo 2 model view LogoutAdmin cho việc đăng xuất và User cho quản lý user

* Trong BaseView LogoutAdmin:
  + Với decorator @login\_required của flask-login ta yêu cầu người dùng phải login rồi mời được truy cập
  + Nếu đã login thì ta sẽ logout người dùng và redirect về url “/admin”
* ModelView User:
  + Hàm is\_accessible() chỉ cho phép người dùng đăng nhập để truy cập
  + Render ra các trường khi hiển thị danh sách người dùng bằng thuọc tính column\_list
  + Hàm on\_model\_change được dùng để mã hoá mật khẩu và insert vào database
  + form\_args để thay đổi input form khi create User mới
  + form\_extra\_field để thêm 1 trường mới thay cho 1 trường đã tồn tại. Ở đây ta thay trường role thay vì ô text ta thay nó bằng 1 select có 2 option là CUSTOMER và ADMIN

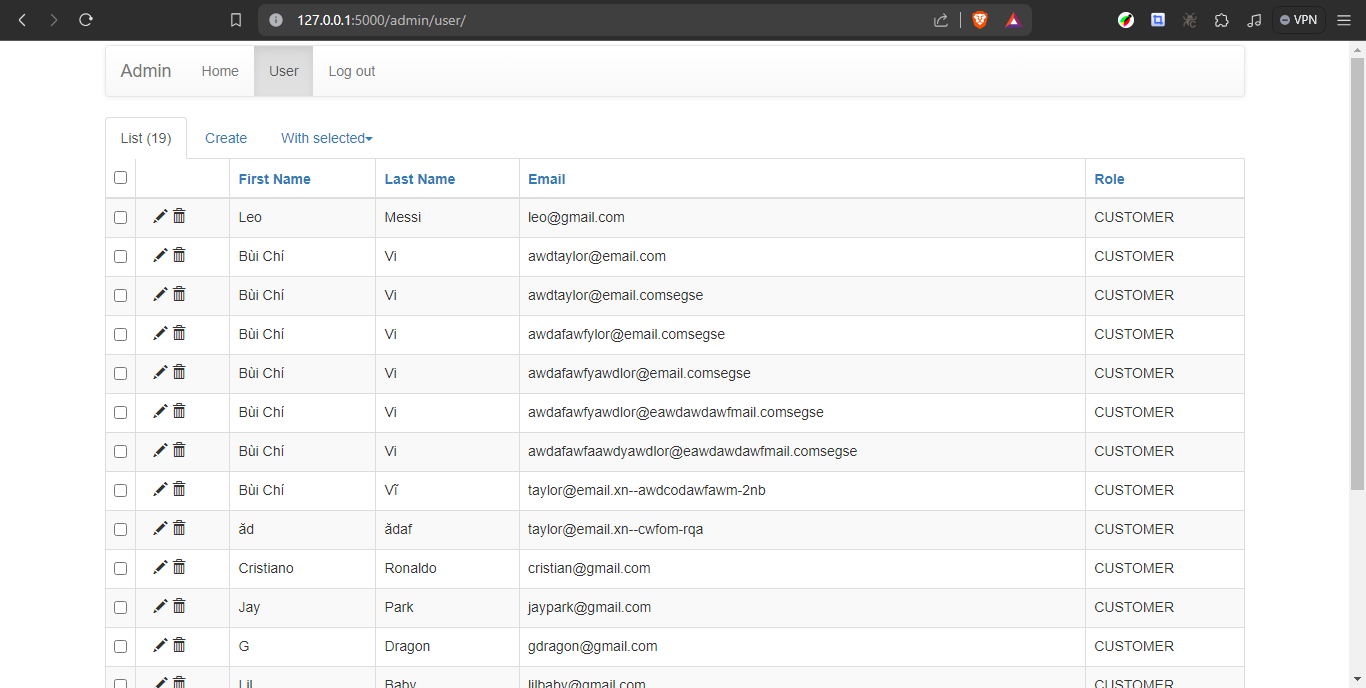
Sau đó ta add vào admin page



Nếu thông tin đăng nhập thoả mãn sẽ cho người dùng truy cập các trường quản trị



Và truy cập vào Model View để quản lý User



Tiếp theo ta xây dựng api đăng nhập cho front end bằng flask-restful

Tạo các Resource

* Login resorce việc đăng nhập nhận email, password và remember và kiểm tra nếu đúng thì login-user lưu người dùng đăng nhập bằng flask-login
* Logout resource đăng xuất người dùng hiện tại
* Sign up resource đăng ký người dùng mới
* Checking Login api để kiểm tra trạng thái hiện tại người dùng có đăng nhập hay không





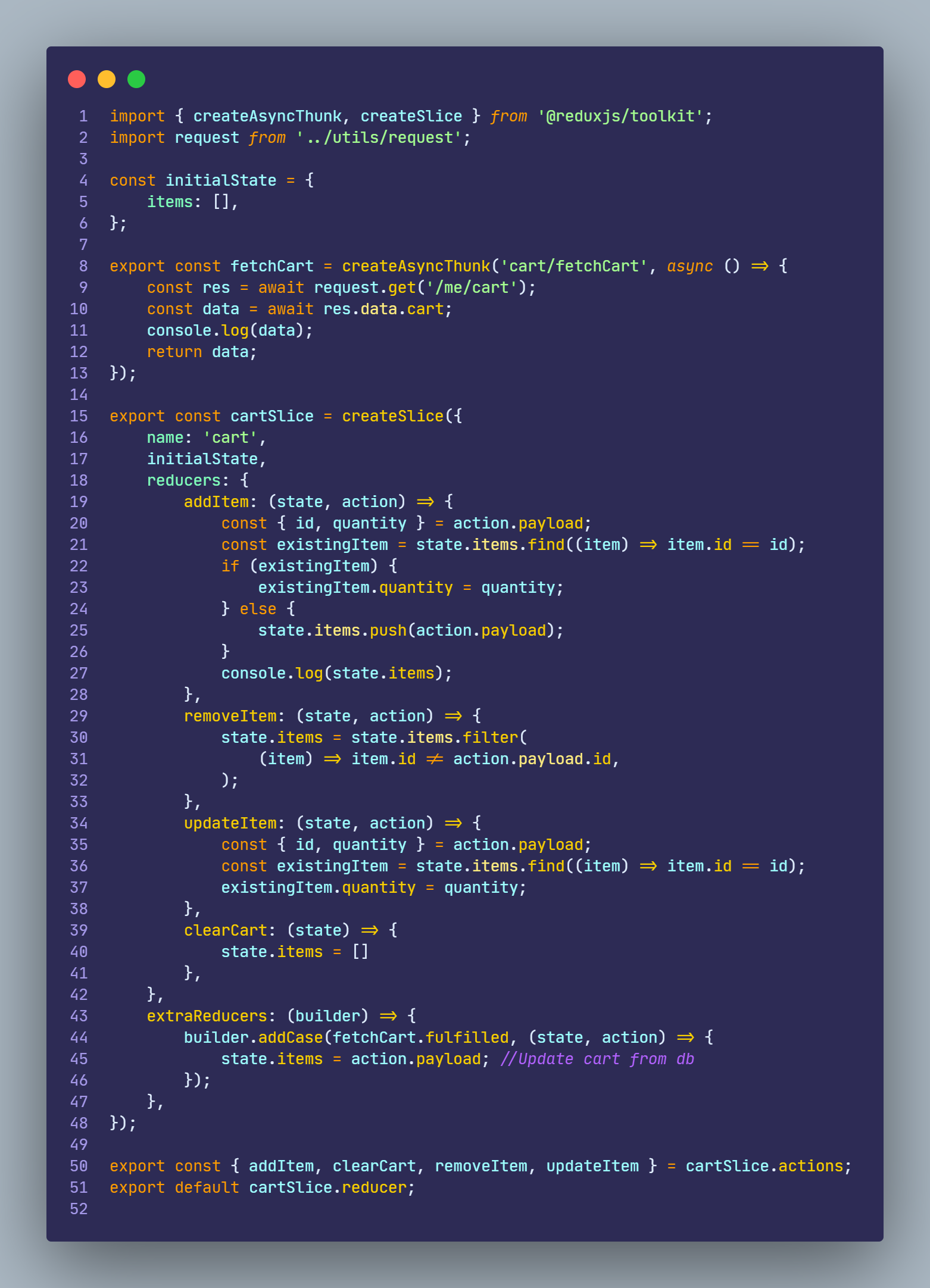




Ở front end ta sử dụng redux toolkit để quản lý trạng thái đăng nhập của user, lưu cart của người dùng

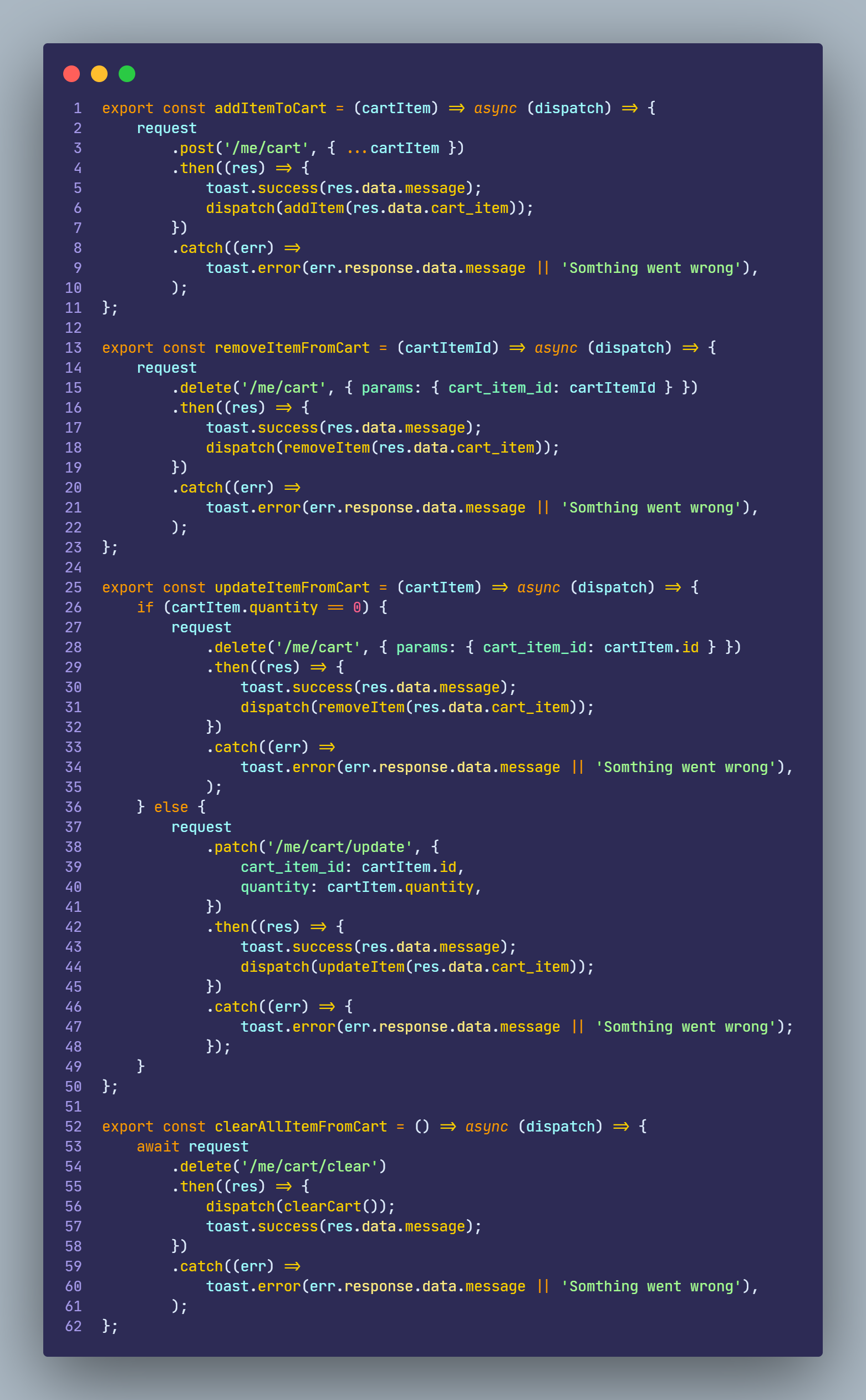
Đầu tiên ta tạo file store/userSlice và store/userCart





* Ở trong file userSlice ta khởi tạo initialState gồm các thông tin:
  + Biến isAuthenticated là ta lấy thông tin của biến isAuthenticated được lưu trong localStorage để khi reload trang nếu đã đăng nhập thì giao diện sẽ dễ quan sát hơn
  + isIntinitalized: dùng để tạo animation khi đăng nhập
  + isAuthenticated: để xác định xem có người dùng đang đăng nhập hay không
  + user: thông tin user
  + Sau đó ta tạo userSlice và các reducer logIn, logOut, checkingLogin, setInitialized để set các thông tin cho state toàn cục
* Tiếp theo trong file cartSlice
  + Ta cũng ta tạo 1 initialState trong đó items là 1 array chứa các cart và hàm fetchCart để lấy dữ liệu cart của người dùng
  + Và tạo các reducer để xử lý thêm, sửa, xoá cart
* Tiếp theo ta tạo file action.js để tạo các action trung gian xử lý yêu cầu api





* Trong folder store tạo file index.js, ta import userSlice và đưa nó vào store



* Tại main.jsx ta bọc toàn bộ app bởi component Provider và truyền store. Điều này cho phép từ bất kỳ component của app ta có thể lấy ra các state. Ngoài ra ta sử dụng thư viện react-toastify để hiển thị các thông báo cần thiết



# TẠO LAYOUT VÀ ROUTES CHO ỨNG DỤNG

* Ta tạo các component như Header, Footer và MainLayout để bọc lấy toàn bộ app đại diện cho layout chính của website



* Tiếp theo ta tạo các route của app



* Biến authRequired trong các route để xác định xem route này có được truy cập mà cần đăng nhập không. Do đó ta tạo 1 component ProtectRoute để bảo vệ các route

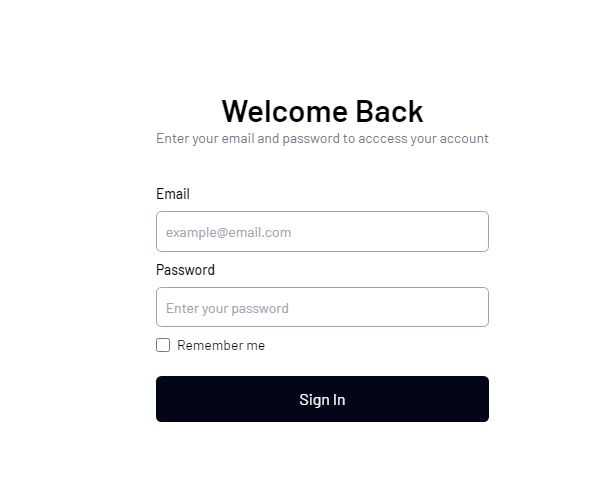


* Sau đó bọc lấy toàn bộ app và render các component theo các route



# XỬ LÝ ĐĂNG NHẬP VÀ ĐĂNG KÝ NGƯỜI DÙNG

* Ta tạo form login cho viẹc đăng nhập của người dùng

****





* Ở đây ta sử dụng thư viện formik và yup để validate form. Khi không có lỗi hàm onSubmit sẽ được gọi. Khi có đủ thông tin ta sẽ dispatch 1 action trung gian logInUser để gọi api đăng nhập và set state user trong redux thành thông tin của người dùng và sau đó hướng người dùng đến trang chủ. Khi đó ta cũng sẽ dispatch action fetchCart để load cart của người dùng hiện tại.
* Tiếp theo là trang đăng ký