Bộ Giáo Dục Và Đào Tạo

**Trường Đại Học Ngoại Ngữ - Tin Học TP-Hồ Chí Minh**

**---o0o---**



**BÁO CÁO KẾT THÚC MÔN HỌC**

***ĐỀ TÀI:* XÂY DỰNG ỨNG DỤNG**

**CHO THUÊ NHÀ ĐẤT DỰA TRÊN HỆ QUẢN TRỊ MYSQL**

Khoa: Công Nghệ Thông Tin

Giảng viên hướng dẫn:  ThS. Lê Thị Minh Nguyện

Sinh viên thực hiện:

Phạm Ngọc Phú MSSV: 19DH110266

Trần Thanh Long MSSV:

Tp. Hồ Chí Minh - Tháng 1/2022

**GIỚI THIỆU**

## Ý nghĩa của đề tài

## Thành phố Hồ Chí Minh là một mảnh đất “màu mỡ” dành cho việc khởi nghiệp và kinh doanh, nơi đây cũng tồn tại nhiều trường đại học danh giá và nổi tiếng. Chính vì thế hằng năm sẽ có một lượng lớn người dân đổ xô về nơi đây để lập nghiệp và các bạn sinh viên đến đây để theo học tại các trường đại học. Bài toán giải quyết cho việc chỗ ở và nơi sinh sống là hết sức khó khăn và nan giải. Chính vì vậy nhóm chúng em đã cũng nhau thiết kế ra ứng dụng quản việc cho thuê nhà đất để đáp ứng cho người mua và người bán những tiện ích hấp dẫn nhất trong việc thuê nhà đất.

## Mục tiêu của đề tài

* Phân tích các yêu cầu của doanh nghiệp .
* Thiết kế dữ liệu phù hợp với yêu cầu của doanh nghiệp.
* Thiết kế xử lý phù hợp với yêu cầu của doanh nghiệp.
* Xây dựng cơ sở dữ liệu và trang web dựa trên các thiết kế.

# KHẢO SÁT HIỆN TRẠNG VÀ XÁC ĐỊNH YÊU CẦU

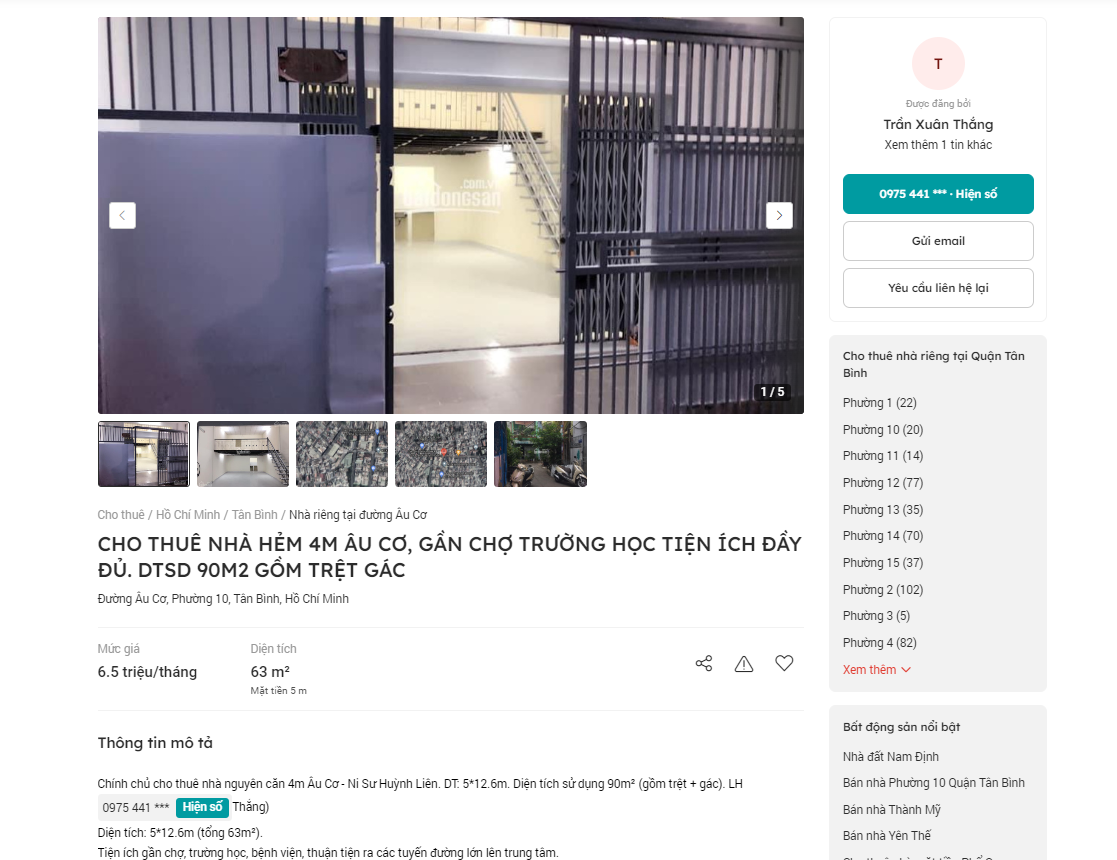
## Khảo sát nghiệp vụ

### Quản lý bài viết cho thuê nhà

Người quản lí quản lí các bài viết của bên cho thuê đăng lên:

* Các bài viết sẽ hiển thị các thông tin cơ bản của nhà được cho thuê để người thuê có thể dễ dàng theo dõi và lựa chọn cho bản thân một nơi ở hợp lí.
* Quản lí có thể theo dõi các bình luận , đánh giá , … từ đó có thể điều chỉnh lại nhà cửa trở nên hoàn hảo và đáp ứng mọi nhu cầu của người thuê.





### Đăng bài viết cho thuê nhà

Người cho thuê sẽ cho thuê nhà bằng cách đăng các bài viết cho thuê nhà lên trên website để cho người thuê dễ dàng theo dõi. Và khi đẫ cho thuê xong thì người cho thuê sẽ xóa bài viết hoặc đánh đấu là đã cho thuê.

Người thuê nhà cũng có thể biết được những căn nhà đang cho thuê bằng cách nghe nhiều người nói lại hay những tờ rơi được dán ở nhiều nơi trên đường phố.



#### Quy trình đăng một bài viết cho thuê nhà

1. Đề ra những nội dung cần thiết cho bài viết, những yêu cầu dành cho người thuê nhà, giá cả,….
2. Đăng bài viết lên website.
3. Đợi xét duyệt.

#### Quy định đăng bài viết, bình luận bài viết và đáh giá cho bài viết.

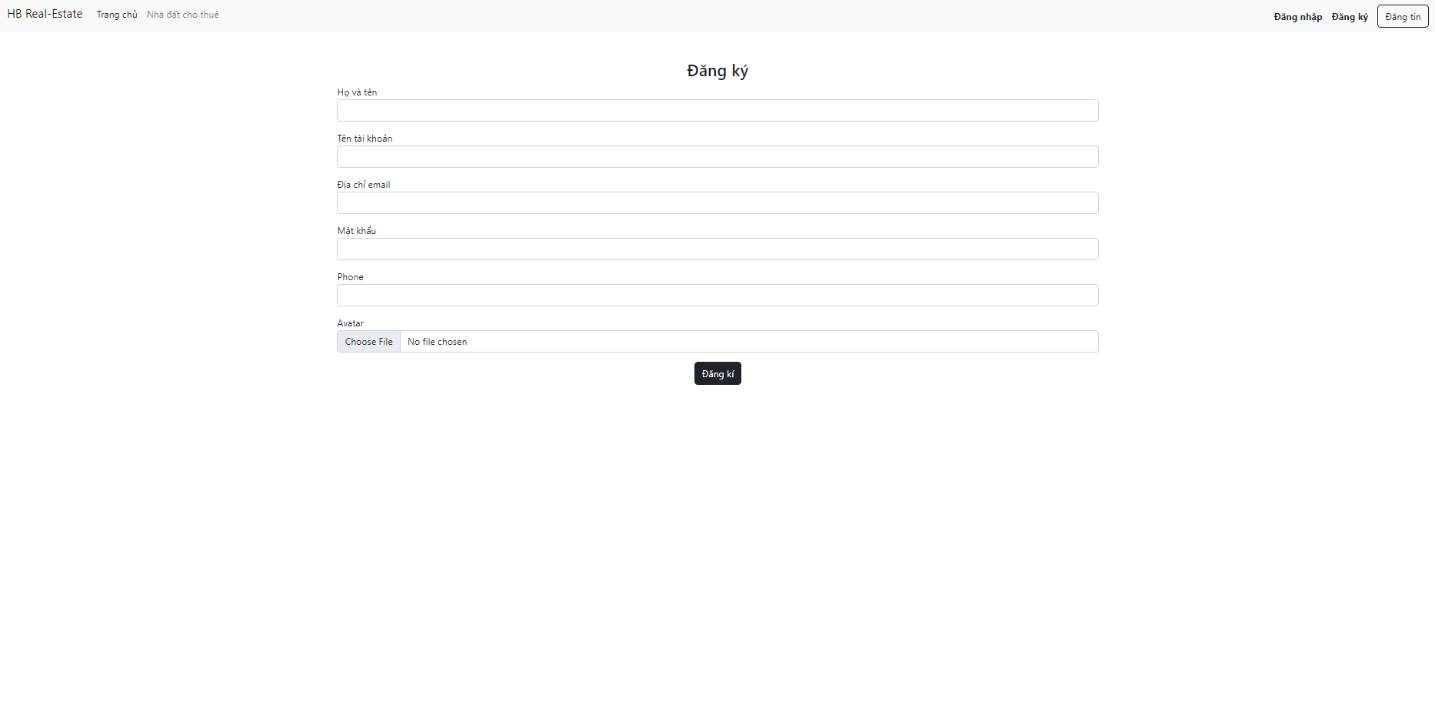
* Khi đăng bài viết thì bên cho thuê không được đăng cùng lúc 2 bài.
* Cần phải có tài khoản đã được đăng kí để có thể đăng bài viết cho thuê nhà trên website.

#### Bình luận bài viết và đáh giá cho bài viết.

* Người thuê có thể đăng bình luận cho bài viết thuê nhà nhưng chỉ được đăng 1 lần cho một bài viết.
* Người dung chỉ có thể báo cáo bài viết ( report ) một lần cho một bài viết.

### Đăng ký

Người dùng có thể đăng ký tài khoản của mình thông qua trang đăng ký của website.



#### Quy trình đăng ký

1. Người dùng vào trang đăng ký tài khoản của webstie.
2. Nhập đầy đủ thông tin và upload hình ảnh của bản than.\
3. Chọn nút đăng ký để đăng ký tài khoản

#### Quy định đăng ký

* Người phải điền đầy đủ các thông tin cá nhân.
* Tên tài khoản đăng ký không được trùng với các tài khoản đã được đăng ký trước đó.
* Mật khẩu phải đầy đủ các ký tự ( chữ hoa, chữ thường, số, ký tự đặc biệt ) và phải từ 8 ký tự trở lên.
* Số điện thoại phải đủ 10 chữ số.
* Địa chỉ email không được trùng với các email của các tài khoản đã được đăng ký và phải đúng định dạng của email.

### Duyệt bài viết

Quản trị hệ thống sẽ xem xét các bài viết có đủ tiêu chuẩn về nhà cho thuê, giá cả, có phù hợp với người thuê hay không. Sau đó sẽ duyệt để đưa lên và người thuê sẽ có thể thấy bài viết đó.

#### Quy trình duyệt bài viết

1. Quản trị hệ thống sau khi đăng nhập sẽ đi tới trang quản lí bài viết.
2. Kiểm tra các thông tin bài viết.
3. Nếu phù hợp thì xác nhận.

#### Quy định duyệt bài viết.

* Không được đăng lại một bài viết khác trùng với bài viết đã được đăng trước đó.
* Phải điền đầy đủ thông tin về nhà thuê, giá cả và các thông tin cụ thể để người thuê có thể yên tâm và dễ hiểu.

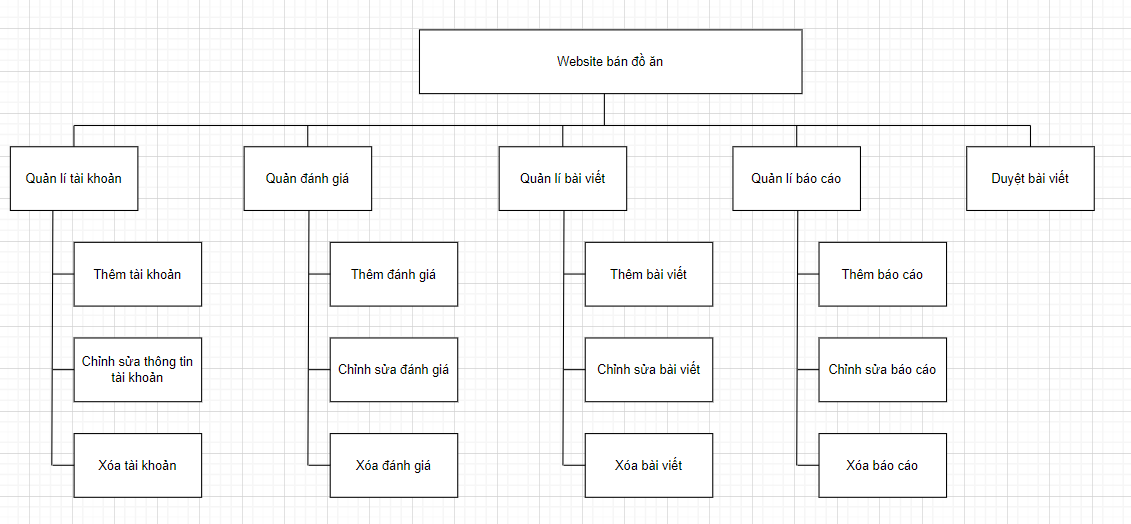
### Báo cáo

Người thuê cảm tháy có vấn đề gì với nơi thuê thì có thể báo cáo bài viết đó để cho các quản trị viên thấy và xem xét lại nơi cho thuê để giúp người thuê tránh khỏi những tình huống không đáng có.

#### Quy định báo cáo

## Yêu cầu

### Yêu cầu chức năng

****

### Yêu cầu phi chức năng

#### Quản lí tài khoản

* Đảm bảo tính bảo mật của hệ thống với người ngoài hệ thống
* Giới hạn các thông tin người dùng được truy cập theo vai trò của người dùng
* Giới hạn các giao tác người dùng có thể thực hiện theo vai trò của người dùng

#### Hiệu suất

* Thời gian phản hồi, thời gian xử lý và thời gian trả kết quả nhanh

#### Tiện dụng

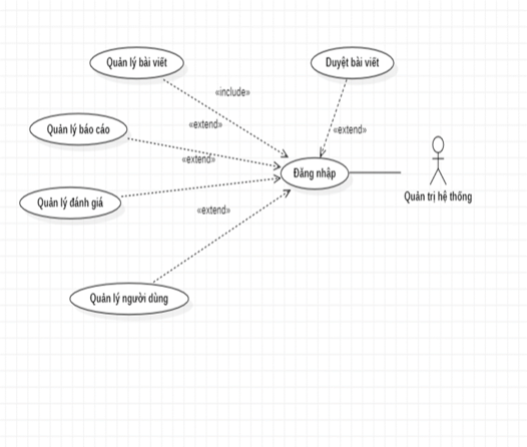
* Cung cấp đầy đủ các chức năng mà người dùng yêu cầu
* Dễ học cách sử dụng
* Hạn chế lỗi nhập liệu

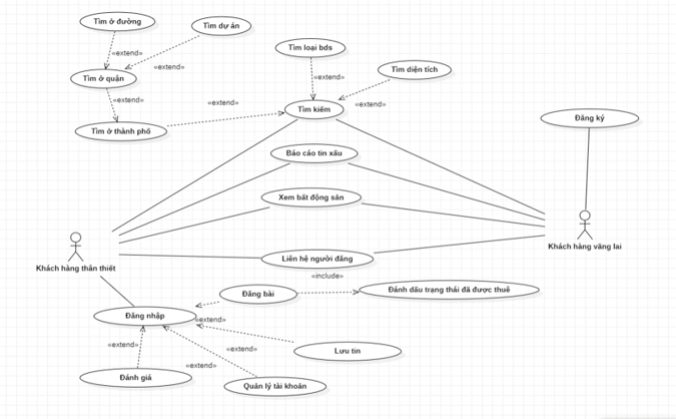
#### An toàn

* Hoạt động ổn định, chính xác
* Có thể khôi phục hệ thống, khôi phục dữ liệu sau các sự cố

# PHÂN TÍCH – THIẾT KẾ DỮ LIỆU

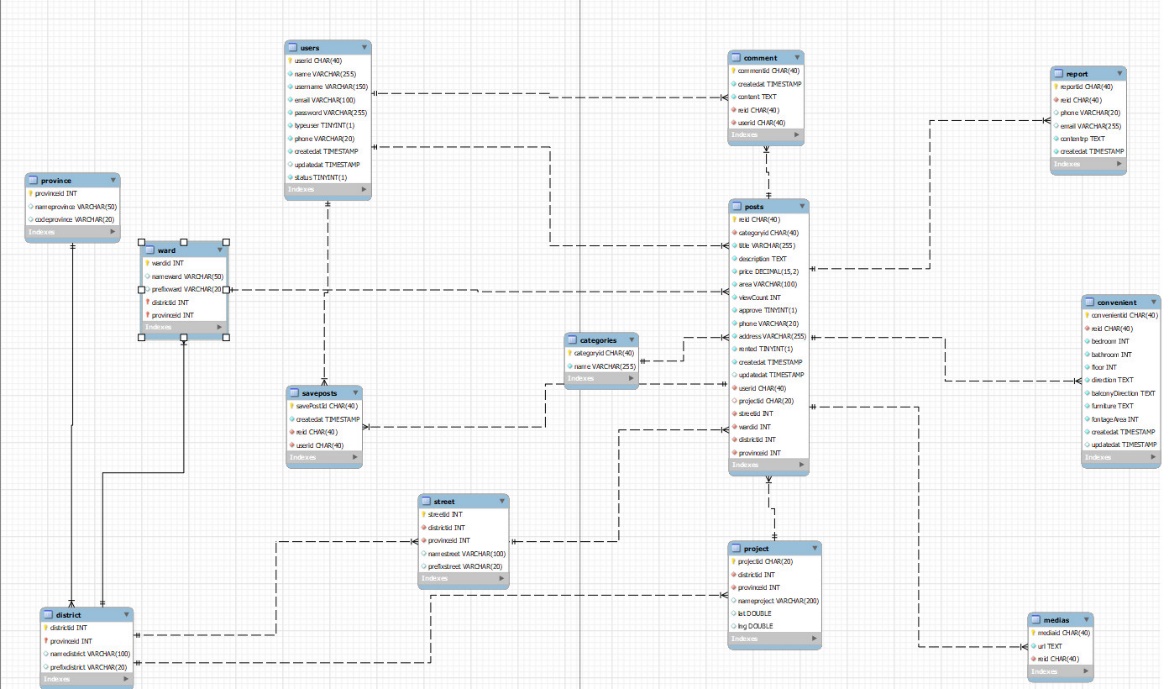
## Phân tích yêu cầu





## Phân tích dữ liệu

### Mô hình thực thể kết hợp



### Mô hình quan hệ

PROVINCE (**PROVINCEID**, nameprovince, codeprovince)

WARD (**WARDID,** nameward, prefixward, districtid, provinceid)

DISTRICT (**DISTRICTID**, provinceid, nameprovince, prefixdistrict)

USER (**USERID**, name, username, password, email, typeuser, phone, createdat, updatedat, status)

SAVEPOSTS (**SAVEPOSTID**, createdat, reid, userid)

STREET (**STREETID**, districtid, provinceid, namestreet, namestreet, prefixstreet)

CATEGORIES (**CATEGORYID**, name)

COMMENT (**COMMENTID**, createdat, content, reid, userid)

PROJECT (**PROJECTID**, districtid, provinceid, nameproject, lat, lng)

POSTS (**POSTID**, categoryid, title, description, price, area, viewcount, approve, phone, address, rented, createdat, updatedat, userid, projected, streetid, wardid, districtid, provinceid)

REPORT (**REPORTID**, reid, phone, email, contentrp, createdat)

MEDIAS (**MEDIAID**, url, reid)

CONVINIENT(**CONVIENTID**,reid,bedroom,bathroom,floor,direction,balconyDirection, furniture, fontageArea, createdat, updatedat)

#### PROVINCE

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **PROVINCEID** | INT | PRIMARY KEY |
| nameprovince | VARCHAR(50) | NOT NULL |
| codeprovince | VARCHAR(20) | NOT NULL |

#### WARD

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **WARDID** | INT | PRIMARY KEY |
| nameward | VARCHAR(50) | NOT NULL |
| prefixward | VARCHAR(20) | NOT NULL |
| districtid | INT | FOREIGN KEY |
| provinceid | INT | FOREIGN KEY |

#### DISTRICT

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **DISTRICTID** | INT | PRIMARY KEY |
| provinceid | INT | FOREIGN KEY |
| nameprovince | VARCHAR(100) | NOT NULL |
| prefixdistrict | VARCHAR(20) | NOT NULL |

#### USER

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **USERID** | CHAR(40) | PRIMARY KEY |
| name | VARCHAR(255) | NOTNULL |
| username | VARCHAR(150) | NOT NULL, UNIQUE |
| password | VARCHAR(150) | NOT NULL |
| email | VARCHAR(100) | NOT NULL, UNIQUE |
| typeuser | TINYINT(1) |  |
| phone | VARCHAR(20) | NOT NULL, UNIQUE |
| createdat | TIMESTAMP |  |
| createdat | TIMESTAMP |  |
| status | TINYINT(1) |  |

#### SAVEPOSTS

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **SAVEPOSTID** | CHAR(40) | PRIMARY KEY |
| createdat | TIMESTAMP |  |
| reid | CHAR(40) |  |
| userid | CHAR(40) |  |

#### STREET

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **STREETID** | INT | PRIMARY KEY |
| districtid | INT | FOREIGN KEY |
| provinceid | INT | FOREIGN KEY |
| namestreet | VARCHAR(100) | NOT NULL |
| prefixstreet | VARCHAR(20) | NOT NULL |

#### CATEGORIES

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **CATEGORYID** | CHAR(40) | PRIMARY KEY |
| name | VARCHAR(225) | NOT NULL |

#### COMMENT

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **COMMENTID** | CHAR(40) | PRIMARY KEY |
| createdat | TIMESTAMP |  |
| content | TEXT |  |
| reid | CHAR(40) | FOREIGN KEY |
| userid | CHAR(40) | FOREIGN KEY |

#### PROJECT

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **PROJECTID** | CHAR(20) | PRIMARY KEY |
| districtid | INT | FOREIGN KEY |
| provinceid | INT | FOREIGN KEY |
| nameproject | VARCHAR(200) | NOT NULL |
| lat | DOUBLE |  |
| lng | BOUBLE |  |

#### POSTS

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **POSTID** | CHAR(40) | PRIMARY KEY |
| categoryid | CHAR(40) | FOREIGN KEY |
| title | VARCHAR(255) | NOT NULL |
| description | TEXT | NOT NULL |
| price | DECIMAL(15,2) | NOT NULL |
| area | VARCHAR(100) | NOT NULL |
| viewcount | INT | NOT NULL |
| approve | TINYINT(1) | NOT NULL |
| phone | VARCHAR(20) | NOT NULL |
| address | VARCHAR(255) | NOT NULL |
| rented | TINYINT(1) | NOT NULL |
| updatedat | TIMESTAMP |  |
| createdat | TIMESTAMP |  |
| userid | CHAR(40) | FOREIGN KEY |
| projected | CHAR(20) |  |
| streetid | INT | NOT NULL |
| wardid | INT | FOREIGN KEY |
| districtid | INT | FOREIGN KEY |
| provinceid | INT | FOREIGN KEY |

#### REPORT

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **REPORTID** | CHAR(40) | PRIMARY KEY |
| reid | CHAR(40) | FOREIGN KEY |
| phone | VARCHAR(20) | NOT NULL, UNIQUE |
| email | VARCHAR(255) | NOT NULL, UNIQUE |
| contentrp | TEXT |  |
| createdat | TIMESTAMP |  |

#### MEDIAS

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **MEDIAID** | CHAR(40) | PRIMARY KEY |
| url | TEXT |  |
| reid | CHAR(40) |  |

#### CONVINIENT

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Ràng buộc / Miền giá trị** |
| **CONVIENTID** | CHAR(40) | PRIMARY KEY |
| reid | CHAR(40) |  |
| bedroom | INT |  |
| bathroom | INT |  |
| floor | INT |  |
| direction | TEXT |  |
| balconyDirection | TEXT |  |
| furniture | TEXT |  |
| fontageArea | INT |  |
| createdat | TIMESTAMP |  |
| updatedat | TIMESTAMP |  |

### Chuẩn hóa mô hình quan hệ

Chú thích : Trong đoạn chuẩn hóa này chúng ta sẽ không đem createdAt, updatedAt vì nó đều là trường dữ liệu bắt buộc trong mỗi bảng

#### USERS

Tập phụ thuộc hàm F = { Userid 🡪 Username, Password, Name,Email,Password  
 , TypeUser, Phone, Status;

Username 🡪 Userid}

🡺 Bảng đạt chuẩn **3NF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa phụ thuộc vào thuộc tính không khóa)

#### POSTS

Tập phụ thuộc hàm F = {Reid 🡪 Categoryid, Title, Description, Price, Area, Viewcount,   
 Approve, Phone, Address, Rented, Userid, Projectid,   
 Streetid, Wardid, Districtid, Provinceid }

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### MEDIAS

Tập phụ thuộc hàm F = { Mediaid 🡪 Url, Reid}

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### CATEGORIES

Tập phụ thuộc hàm F = { Categoryid 🡪 Name }

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### CONVENIENT

Tập phụ thuộc hàm F = { Convenientid, Reid 🡪 Bedroom, Bathroom, Floor, Direction,

Balconydirection, Funiture, FontageArea }

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### REPORT

Tập phụ thuộc hàm F = { Reportid 🡪 Reid, Phone, Email, Contentrp}

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### SAVEPOSTS

Tập phụ thuộc hàm F = {Savepostid 🡪 Reid, Userid}

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### COMMENT

Tập phụ thuộc hàm F = { Commentid 🡪 Content, Reid, Userid }

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### PROVINCE

Tập phụ thuộc hàm F = { Provinceid 🡪 Nameprovince, Codeprovince }

🡺 Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### DISTRICT

Tập phụ thuộc hàm F = { Districtid, Provinceid 🡪 Namedistrict, Prefixdistrict }

🡺Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### WARD

Tập phụ thuộc hàm F = { Wardid, Districtid, Provinceid 🡪 Nameward, Prefixward }

🡺Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### STREET

Tập phụ thuộc hàm F = { Streetid, Districtid, Provinceid 🡪 Namestreet, Prefixstreet }

🡺Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

#### PROJECT

Tập phụ thuộc hàm F = { Projectid, Districtid, Provinceid 🡪 Nameproject, Prefixproject }

🡺Bảng đạt chuẩn **BCNF** (Các thuộc tính không khóa phụ thuộc hoàn toàn vào thuộc tính khóa và không phụ thuộc bắc cầu vào thuộc tính không khóa khác; Thuộc tính khóa không phụ thuộc vào thuộc tính không khóa)

## Ràng buộc trọn vẹn

### Username không được trùng

* Bối cảnh: USERS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USERS** | + | - | + (Username) |

### Password không được để trống

* Bối cảnh: USERS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USERS** | + | - | + (Password) |

### Typeuser chỉ có thể là 0 hoặc 1

* Bối cảnh: USERS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USERS** | +(typeuser) | - | + (typeuser) |

### Status (thể hiện trạng thái chặn) chỉ có thể là 0 hoặc 1

* Bối cảnh: USERS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USERS** | +(status) | - | + (status) |

### Typeuser chỉ có thể là 0 hoặc 1

* Bối cảnh: USERS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USERS** | +(typeuser) | - | + (typeuser) |

### Thông tin địa chỉ không được sai

* Bối cảnh: POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **POSTS** | +(provinceid,wardid,districtid,projectid) | + | - |

### Số lượng bài đăng trong tháng không được quá 5 bài

* Bối cảnh: POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **POSTS** | +(reid) | + | - |

### Giá cho thuê nhà đất trên 100.000 VND

* Bối cảnh: POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **POSTS** | +(price) | + | - |

### Định dạng số điện thoại đúng

* Bối cảnh: POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **POSTS** | +(phone) | + | - |

### Lưu 1 bài đăng không được yêu thích quá 2 lần

* Bối cảnh: SAVEPOSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| SAVEPOSTS | +(reid,userid) | - | +(reid,userid) |

### Khi lưu bài sẽ tăng lượt yêu thích bài post

* Bối cảnh: SAVEPOSTS, POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| SAVEPOSTS | + (reid,userid) | -(reid,userid) | - |
| POSTS | + | + | +(viewcount) |

### Khi bỏ lưu bài sẽ giảm lượt yêu thích bài post

* Bối cảnh: SAVEPOSTS, POST
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| SAVEPOSTS | + (reid,userid) | -(reid,userid) | - |
| POSTS | + | + | +(viewcount) |

### Bình luận không được bình luận hai lần trong cùng một bài viết

* Bối cảnh: COMMENT
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **COMMENT** | + (commentid,reid) | - | +  (commentid,reid) |

### Không được đăng bài viết giống nhau nếu muốn đăng thì phải đổi thuộc tính userid thành NULL.

* Bối cảnh: POSTS, USER
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **USER** | - | + | - (\*) |
| **POSTS** | - | - | + (userid) |

### Ngày đăng bài phải nhỏ hơn ngày nhà được cập nhật lại thành được thuê

* Bối cảnh: POSTS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **POSTS** | + | - | + (createdat, updatedat) |

### Khi tố cáo bài đăng rồi không được tố cáo lần nữa dựa trên email

* Bối cảnh: REPORT
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **REPORT** | +(email) | - | + (email) |

### Khi tố cáo bài đăng rồi không được tố cáo lần nữa dựa trên số điện thoại

* Bối cảnh: REPORT
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **REPORT** | +(phone) | - | + (phone) |

### Thêm hình ảnh, videos của bài đăng không quá 5 tấm ảnh

* Bối cảnh: MEDIAS
* Bảng tầm ảnh hưởng:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Thêm** | **Xóa** | **Sửa** |
| **MEDIAS** | +(reid,mediasid) | - | - |

# THIẾT KẾ XỬ LÝ TRONG CLIENT/ SERVER

## Các giao tác

### Thêm tài khoản

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server tạo tài khoản và truyền tham số |  |
|  | **Tạo tài khoản**   * Tạo ra mã khóa cho user dựa trên mã hóa uuid * Kiểm tra username có tồn tại * Tạo function mã hóa mật khẩu * Khai báo biến và gọi function mã hóa mật khẩu gán giá trị cho biến đã vừa khai báo * Thực thi chèn thông tin dữ liệu vào bảng USERS * Thêm user vào Role thích hợp * Cấp TSK\_Client (tài khoản khách) quyền impersonate user |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 38 Các bước xử lý giao tác Thêm tài khoản

### Đăng nhập

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server xác thực user, trả về dữ liệu, truyền tham số |  |
|  | **Đăng nhập**   * Kiểm tra username có tồn tại * Kiểm tra mật khẩu * Đọc dữ liệu từ bảng USERS |
| Nhận kết quả giao tác lưu vào trong session |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 39 Các bước xử lý giao tác Đăng nhập

### Tố cáo

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server lưu dữ liệu tố cáo |  |
|  | **Tố cáo**   * Tạo function kiểm tra dữ liệu name, phone * Kiểm tra độ dài content * Kiểm tra người dùng đã gửi tố cáo qua chưa * Thêm dữ liệu vào bảng REPORT |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 40 Tố cáo

### Cập nhật trạng thái đã thuê cho bài đằng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật trạng thái bài đăng |  |
|  | **Tố cáo**   * Kiểm tra bài post có bị khóa * Kiểm tra trạng thái đã được cập nhật trước đó chưa * Kiểm tra có phải chủ nhân bài đăng * Kiểm tra bài đăng có tồn tại * Cập nhật dữ liệu trong bảng POSTS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 41 Tố cáo

### Xóa tài khoản

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server xóa tài khoản và truyền tham số |  |
|  | **Xóa tài khoản**   * Xóa tài khoản khỏi bảng TAIKHOAN * Hủy quyền impersonate user được xóa của TSK\_Client (tài khoản khách) * Xóa user trong database |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 39 Các bước xử lý giao tác Xóa tài khoản

### Thêm lưu bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server lưu bài đăng tài khoản |  |
|  | **Lưu bài đăng**   * Kiểm tra bài đăng đã được lưu trước đó chưa * Kiểm tra bài đăng có tồn tại * Thêm dữ liệu vào bảng SAVEPOSTS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 40 Lưu bài đăng

### Xóa lưu bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server hủy lưu bài đăng |  |
|  | **Hủy lưu bài đăng**   * Kiểm tra bài đăng đã được lưu chưa * Kiểm tra bài đăng có tồn tại * Xóa dữ liệu khỏi bảng SAVEPOSTS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 41 Hủy lưu bài đăng

### Cập nhật thông tin tài khoản

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật thông tin tài khoản |  |
|  | **Cập nhật thông tin tài khoản**   * Kiểm tra tài khoản tồn tại * Cập nhật dữ liệu tại bảng USERS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 42 Các bước xử lý giao tác Xóa giáo viên

### Thêm bình luận ở bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server thêm bình luận |  |
|  | **Thêm bình luận bài đăng**   * Kiểm tra bài đăng có tồn tại * Kiểm tra tài khoản có tồn tại * Kiểm tra nội dụng bình luận * Kiểm tra đã bình luận trước đó ở bài đăng * Thêm dữ liệu vào bảng Comment |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 43 Các bước xử lý giao tác Cập nhật giá

### Thêm bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server thêm bài đăng |  |
|  | **Thêm ca học**   * Kiểm tra tài khoản người đăng * Kiểm tra địa chỉ hợp lệ * Kiểm tra loại nhà đất có hợp lệ * Kiểm tra số hình chỉ tối đa 5 tấm * Kiểm tra dữ liệu tiện ích đã được thêm trước đó chưa * Tạo mã key cho bài POSTS bằng uuid() * Thêm dữ liệu vào bảng POSTS * Thêm dữ liệu bào bảng CONVENIENT * Thêm dữ liệu vào bảng MEDIAS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 44 Thêm bài đăng

### Cập nhật trạng thái chặn bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server chặn bài đăng |  |
|  | **Chặn bài đăng**   * Kiểm tra bài đăng tồn tại * Kiểm tra có đủ quyền để chặn * Kiểm tra đã từng chặn * Cập nhật trạng thái chặn bài đăng |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 45 Cập nhật trạng thái chặn của bài đăng

### Cập nhật trạng thái gỡ chặn bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật ca học và truyền tham số |  |
|  | **Cập nhật trạng thái gỡ chặn**   * Kiểm tra bài đăng tồn tại * Kiểm tra có đủ quyền để chặn * Kiểm tra đã từng chặn * Cập nhật trạng thái gỡ chặn bài đăng |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 46 Các bước xử lý giao tác Cập nhật trạng thái gỡ chặn bài đăng

### Xóa bài đăng

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật ca học và truyền tham số |  |
|  | **Xóa bài đăng**   * Kiểm tra bài đăng tồn tại * Kiểm tra có đủ quyền để xóa * Xóa dữ liệu từ bảng MEDIAS * Xóa dữ liệu từ bảng POSTS |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 46 Các bước xử lý giao tác Cập nhật ca học

### Cập nhật trạng thái chặn tài khoản

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật trạng thái chặn tài khoản |  |
|  | **Cập nhật trạng thái chặn tài khoản**   * Kiểm tra tài khoản tồn tại * Kiểm tra có đủ quyền để chặn * Kiểm tra đã được trạng thái đã được chặn chưa * Cập nhật trạng thái chặn tài khoản |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 46 Các bước xử lý giao tác Cập nhật trạng thái chặn tài khoản

### Cập nhật trạng thái gỡ chặn tài khoản

|  |  |
| --- | --- |
| **Client** | **Server** |
| Yêu cầu server cập nhật trạng thái gỡ chặn tài khoản |  |
|  | **Cập nhật trạng thái gỡ chặn tài khoản**   * Kiểm tra tài khoản tồn tại * Kiểm tra có đủ quyền để chặn * Kiểm tra đã được chặn trước đó chưa * Cập nhật trạng thái gỡ chặn tài khoản |
| Nhận kết quả giao tác |  |
| Thông báo kết quả giao tác cho người dùng |  |

Bảng 46 Các bước xử lý giao tác Cập nhật trạng thái gỡ chặn tài khoản

## Phân quyền

### Các đối tượng phân quyền

* Quản lý: Quản lý các chức năng trong hệ thống
* Khách hàng thân thiết: các khách hàng chưa có tài khoản
* Khách hàng vãng lai: các khách hàng đã có tài khoản

### Phân quyền trên các bảng

*Ghi chú:* S – Select, I – Insert, U – Update, D – Delete

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Khách hàng vãng lai** | **Khách hàng thân thiết** | **Quản lý** |
| **CATEGORIES** | S | S | S, I, U, D |
| **COMMENT** | S | S, I, U, D | S |
| **CONVENIENT** | S | S, I | S |
| **DISTRICT** | S | S | S, I, U, D |
| **MEDIAS** | S | S, I | S |
| **POSTS** | S | S, I, D | S, U |
| **PROJECT** | S | S, I, D | S |
| **PROVINCE** | S | S, I, D | S |
| **REPORT** | S | S, I, U, D | S |
| **SAVEPOSTS** | S | S, I, D | S |
| **STREET** | S | S, I, D | S |
| **USERS** | S | S, U | S, I, U |
| **WARD** | S | S, I, D | S, I, U, D |

Bảng 60 Phân quyền trên các bảng

### Phân quyền trên các giao tác

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Quản lý** | **Khách hàng vãng lai** | **Khách hàng than thiêt** |
| **Đăng ký tài khoản** | X | X | X |
| **Đăng nhập tài khoản** | X | X |  |
| **Chặn tài khoản** | X |  |  |
| **Sữa tài khoản** |  |  | X |
| **Tìm kiếm từ khóa** | X | X | X |
| **Tìm kiếm địa chỉ** | X | X | X |
| **Xem chi tiết nhà đất** | X | X | X |
| **Yêu thích** | X |  | X |
| **Bỏ yêu thích** | X |  | X |
| **Tố Cáo** | X |  | X |
| **Báo cáo tin xấu** | X | X | X |
| **Sữa thông tin tài khoản** | X |  | X |
| **Xem bài đăng đã đăng** | X |  | X |
| **Xóa bài đăng** | X |  | X |
| **Chặn bài đăng** | X |  |  |
| **Cập nhật trạng thái bài đăng** | X |  | X |
| **Xem tin đã yêu thích** | X |  | X |
| **Đăng bài** | X |  | X |
| **Chặn tài khoản** | X |  |  |
| **Mở chặn bài đăng** | X |  |  |
| **Mở chặn tài khoản** | X |  |  |
| **Xóa bài đăng** | X |  | X |

## Backup

* Full database backup: Mỗi chủ nhật hàng tuần vào lúc 23 giờ.
* Differenitial backup: Mỗi thứ 3, thứ 5, thứ 7 hàng tuần vào lúc 12 giờ
* Logs backups: Mỗi giờ vào phút 5, 15, 25, 35, 45, 55.

# TRIỂN KHAI THỰC NGHIỆM

## Phía Server

### Ràng buộc toàn vẹn

#### Username không được trùng

DROP TRIGGER IF EXISTS tg\_after\_username;

DELIMITER $$

CREATE TRIGGER tg\_after\_username

BEFORE INSERT

ON users FOR EACH ROW

BEGIN

    DECLARE is\_exists\_username INT DEFAULT -1;

    DECLARE msg VARCHAR(200);

    SELECT COUNT(\*) INTO is\_exists\_username

    FROM  users U

    WHERE U.username = new.username;

    SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_after\_username: ', cast(new.username as char));

    IF is\_exists\_username > 0  THEN

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Password không được để trống

DROP TRIGGER IF EXISTS tg\_before\_validate\_password;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_password

BEFORE INSERT

ON users FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_after\_username: ', cast(new.password as char));

    IF LENGTH(new.password) < 7  THEN

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

 DELIMITER ;

#### TypeUser chỉ được 0 hoặc 1

DROP TRIGGER IF EXISTS tg\_before\_type\_user;

DELIMITER $$

CREATE TRIGGER tg\_before\_type\_user

BEFORE INSERT

ON users FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    IF  NEW.typeuser != 0 AND NEW.typeuser != 1 THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_before\_type\_user: ', cast(new.typeuser as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Status chỉ được 0 hoặc 1

DROP TRIGGER IF EXISTS tg\_before\_status\_user;

DELIMITER $$

CREATE TRIGGER tg\_before\_status\_user

BEFORE INSERT

ON users FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    IF NEW.status <> 0  AND NEW.status <> 1 THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_before\_type\_user: ', cast(new.status as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Địa chỉ phải chính xác

DROP TRIGGER IF EXISTS tg\_before\_validate\_post;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_post

BEFORE INSERT

ON posts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    DECLARE check\_address  INT DEFAULT -1;

    SELECT  COUNT(\*) INTO check\_address

    FROM province p JOIN (SELECT districtid, provinceid

                         FROM district) d

                    ON d.provinceid = p.provinceid

                    JOIN (SELECT wardid, districtid

                          FROM ward ) w

                    ON  w.districtid = d.districtid

                    JOIN (SELECT streetid, districtid

                          FROM street) s

                    ON  s.districtid = w.districtid

    WHERE p.provinceid = new.provinceid AND d.districtid = new.districtid

         AND w.wardid = new.wardid AND s.streetid = new.streetid;

    IF check\_address <= 0 THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value address in tg\_before\_validate\_post: ', cast(new.provinceid as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Gía tiền bài đăng phải trên 100.00 VND

DROP TRIGGER IF EXISTS tg\_before\_validate\_post;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_post

BEFORE INSERT

ON posts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    IF new.price < 100   THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative price  in tg\_before\_validate\_post: ', cast(new.price as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Số điện thoại phải 9 số hoặc 10 số

DROP TRIGGER IF EXISTS tg\_before\_validate\_post;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_post

BEFORE INSERT

ON posts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    IF length(new.phone) < 9 OR length(new.phone) > 10 THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value  phone in tg\_before\_validate\_post: ', cast(new.phone as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Số lượng đăng bài trong tháng chỉ được 5 bài

DROP TRIGGER IF EXISTS tg\_before\_validate\_post;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_post

BEFORE INSERT

ON posts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    DECLARE count\_posts INT DEFAULT -1;

    SELECT COUNT(p.reid) INTO count\_posts

    FROM posts p

    WHERE MONTH(p.createdat) = month(CURRENT\_DATE()) AND p.userid = new.userid;

    IF count\_posts > 5 THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value count post in tg\_before\_validate\_post: ', cast(count\_posts as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

DELIMITER ;

#### Lưu bài đăng chỉ được lưu 1 lần

DROP TRIGGER IF EXISTS tg\_before\_validate\_save\_user;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_save\_user

BEFORE INSERT

ON saveposts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    DECLARE is\_user\_save INT DEFAULT -1;

    SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_before\_validate\_save\_user: ', cast(new.userid as char));

    SELECT COUNT(\*) INTO is\_user\_save

    FROM saveposts

    WHERE reid = new.reid AND  userid = new.userid;

    IF  is\_user\_save > 0  THEN

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

 DELIMITER ;

#### Tố cáo với 1 email hoặc số điện thoại chỉ được 1 lần cho 1 bài đăng

DROP TRIGGER IF EXISTS tg\_before\_validate\_save\_user;

DELIMITER $$

CREATE TRIGGER tg\_before\_validate\_save\_user

BEFORE INSERT

ON saveposts FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    DECLARE is\_user\_save INT DEFAULT -1;

    SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_before\_validate\_save\_user: ', cast(new.userid as char));

    SELECT COUNT(\*) INTO is\_user\_save

    FROM saveposts

    WHERE reid = new.reid AND  userid = new.userid;

    IF  is\_user\_save > 0  THEN

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

 DELIMITER ;

#### Số lượng thêm hình cho 1 bài đăng không quá 5 tấm

DROP TRIGGER IF EXISTS tg\_before\_limit\_insert\_medias;

DELIMITER $$

CREATE TRIGGER tg\_before\_limit\_insert\_medias

BEFORE INSERT

ON medias FOR EACH ROW

BEGIN

    DECLARE msg VARCHAR(200);

    DECLARE quantity\_medias INT DEFAULT -1;

    SET  quantity\_medias = (SELECT COUNT(\*)

                           FROM  medias

                           WHERE reid = new.reid);

    IF quantity\_medias = 5  THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_before\_limit\_insert\_medias: ',cast(new.reid as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

END $$

 DELIMITER ;

#### Khi yêu thích 1 bài đăng số lượt yêu thích của bài đăng phải tăng lên

DROP TRIGGER IF EXISTS tg\_after\_auto\_update\_viewcount;

DELIMITER $$

CREATE TRIGGER tg\_after\_auto\_update\_viewcount

after INSERT

ON saveposts FOR EACH ROW

BEGIN

    DECLARE reid\_need\_update CHAR (40) DEFAULT NULL;

    DECLARE msg VARCHAR(200);

    DECLARE quantity\_save INT DEFAULT -1;

    SET reid\_need\_update = (SELECT reid

                            FROM posts

                            WHERE reid = new.reid);

    IF reid\_need\_update IS NULL OR reid\_need\_update = '' THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_after\_auto\_save\_user: ',cast(new.reid as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

    SET quantity\_save = (SELECT viewCount

                         FROM posts

                         WHERE reid = reid\_need\_update);

    SET SQL\_SAFE\_UPDATES = 0;

        UPDATE posts

        SET viewCount = quantity\_save + 1

        WHERE reid = reid\_need\_update;

    SET SQL\_SAFE\_UPDATES = 1;

END $$

DELIMITER ;

#### Khi 1 yêu thích bỏ đi thì giảm lượt yêu thích bài đăng

DROP TRIGGER IF EXISTS tg\_after\_auto\_update\_reduce\_viewcount;

DELIMITER $$

CREATE TRIGGER tg\_after\_auto\_update\_reduce\_viewcount

before DELETE

ON saveposts FOR EACH ROW

BEGIN

    DECLARE reid\_need\_update CHAR (40) DEFAULT NULL;

    DECLARE msg VARCHAR(200);

    DECLARE quantity\_save INT DEFAULT -1;

    SET reid\_need\_update = (SELECT reid

                            FROM saveposts

                            WHERE userid = old.userid AND reid = old.reid);

    IF reid\_need\_update IS NULL OR reid\_need\_update = '' THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_after\_auto\_update\_reduce\_viewcount: ',cast(old.reid as char));

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

    SET quantity\_save = (SELECT viewCount

                         FROM posts

                         WHERE reid = reid\_need\_update) - 1;

    IF quantity\_save < 0  THEN

        SET msg =  concat('MyTriggerError: Trying to insert a negative value in tg\_after\_auto\_update\_reduce\_viewcount quantity not right ');

        SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = msg;

    END IF;

    SET SQL\_SAFE\_UPDATES = 0;

        UPDATE posts

        SET viewCount = quantity\_save

        WHERE reid = reid\_need\_update;

    SET SQL\_SAFE\_UPDATES = 1;

END $$

 DELIMITER ;

### Các chức năng

#### Đăng ký

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_SHAPassword $$

CREATE FUNCTION fnc\_SHAPassword(pr\_password varchar(50), pr\_private\_key varchar(50) )

RETURNS VARCHAR(50)

DETERMINISTIC

begin

    return sha1(concat(pr\_password, pr\_private\_key));

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Register $$

CREATE PROCEDURE sp\_Register(pr\_username varchar(50), pr\_password varchar(50),pr\_name varchar(150),pr\_email varchar(100),pr\_phone varchar(11))

BEGIN

    DECLARE isExists int DEFAULT -1;

    DECLARE pw varchar(50);

    DECLARE privateKey text DEFAULT uuid();

    SELECT COUNT(\*) INTO isExists

    FROM users U

    WHERE U.username = pr\_username;

    IF (isExists>0) THEN

        SELECT 0;

    ELSE

        SET pw= fnc\_SHAPassword(pr\_password, privateKey);

        START TRANSACTION;

            SET SQL\_SAFE\_UPDATES = 0;

            INSERT INTO USERS(userid ,name,username,email,password,phone)

            VALUES (privateKey,pr\_name,pr\_username,pr\_email,pw,pr\_phone);

            SET SQL\_SAFE\_UPDATES = 1;

        COMMIT;

        SELECT 1 ;

    END IF;

END; $$

#### Đăng nhập

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_SHAPassword $$

CREATE FUNCTION fnc\_SHAPassword(pr\_password varchar(50), pr\_private\_key varchar(50) )

RETURNS VARCHAR(50)

DETERMINISTIC

begin

    return sha1(concat(pr\_password, pr\_private\_key));

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Login $$

CREATE PROCEDURE sp\_Login(pr\_username varchar(50), pr\_password varchar(50))

BEGIN

    DECLARE private\_key VARCHAR(50);

    DECLARE isExists INT DEFAULT -1;

    DECLARE pw VARCHAR(50)  ;

    SELECT  u.userid INTO private\_key  FROM users u WHERE u.username= pr\_username;

    SET pw=fnc\_SHAPassword(pr\_password, private\_key); #function này đã tạo ra ở bước trên

    SELECT COUNT(\*) INTO isExists FROM users u WHERE u.username=pr\_username AND u.password=pw;

    IF(isExists >0) THEN

        SELECT u.name,  u.username,u.userid,

               u.email, u.phone

        FROM users u

        where u.userid = private\_key;

    ELSE

        SELECT 0;

    END IF;

END; $$

#### Tố cáo

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_checkNamePhoneReport $$

CREATE FUNCTION fnc\_checkNamePhoneReport(pr\_reid varchar(40), pr\_phone varchar(40),pr\_email varchar(40))

RETURNS INT

DETERMINISTIC

begin

    DECLARE result INT DEFAULT -1 ;

    SELECT COUNT(\*) INTO result

    FROM report r

    WHERE r.reid = pr\_reid AND r.phone = pr\_phone AND r.email = pr\_email;

    IF (result > 0 ) THEN

        SET result = 1;

    ELSE

        SET result = 0;

    END IF;

    RETURN result;

END; $$

#### Cập nhật trạng thái bài đăng

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_update\_status\_post $$

CREATE PROCEDURE sp\_update\_status\_post(pr\_id\_user char(40), pr\_id\_post char(40))

BEGIN

    DECLARE is\_exists INT DEFAULT -1;

    DECLARE check\_blocked INT DEFAULT -1;

    DECLARE is\_update\_status INT DEFAULT -1 ;

    DECLARE is\_belong\_user INT DEFAULT -1 ;

    SET is\_exists  = fnc\_is\_exists\_posts(pr\_id\_post);

    IF is\_exists > 0 THEN

        SET check\_blocked = fnc\_check\_blocked(pr\_id\_post);

        IF check\_blocked > 0 THEN

            SELECT 0 as "status", "Bài post đã bị khóa" as "message", '' as 'data';

        ELSE

            SELECT COUNT(\*) INTO is\_update\_status

            FROM posts p

            WHERE p.reid = pr\_id\_post AND p.rented = 0 ;

            IF is\_update\_status > 0 THEN

                SELECT COUNT(\*) INTO is\_belong\_user

                FROM posts p

                WHERE p.reid = pr\_id\_post AND p.userid = pr\_id\_user;

                IF is\_belong\_user > 0 THEN

                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

                    START TRANSACTION;

                        SET SQL\_SAFE\_UPDATES = 0;

                            UPDATE Posts

                            SET rented = 1

                            WHERE reid = pr\_id\_post;

                        SET SQL\_SAFE\_UPDATES = 1;

                    COMMIT;

                    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

                ELSE

                    SELECT 0 as 'status', "Bài post này không thuộc quyền sở hữu của bạn" as "message",

                                            '' as 'data';

                END IF;

            ELSE

                SELECT 0 as status, "Bài post đã được cập nhật trạng thái" as message , '' as 'data';

            END IF;

        END IF;

    ELSE

        SELECT 0 as 'status', 'Bài Posts không tồn tại' as 'message';

    END IF;

END; $$

#### Chi tiết bài đăng

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_show\_detail\_info $$

CREATE PROCEDURE sp\_show\_detail\_info(pr\_reid char(40))

BEGIN

    DECLARE isExists int DEFAULT -1;

    DECLARE isExistsProjectid VARCHAR(40) DEFAULT NULL;

    SELECT COUNT(\*) INTO isExists

    FROM posts p

    WHERE p.reid = pr\_reid;

    SELECT projectid INTO isExistsProjectid

    FROM posts p

    WHERE p.reid = pr\_reid ;

    IF (isExists = 0) THEN

        SELECT 'Không có tin này!';

    ELSE

        IF ISNULL(isExistsProjectid) = 1 THEN

            SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                        SELECT p.reid, p.title, p.description, p.price

                            , p.area, p.viewCount, p.phone

                            , p.address, p.rented, p.createdat

                            , c.name as 'categories', pr.nameprovince

                            , dis.namedistrict, w.nameward,st.namestreet as 'street'

                            , u.name, u.email, u.phone

                     FROM posts p JOIN users u

                                  ON u.userid = p.userid

                                  JOIN categories c

                                  ON c.categoryid = p.categoryid

                                  JOIN (SELECT provinceid,nameprovince

                                        FROM province ) pr

                                  ON pr.provinceid = p.provinceid

                                  JOIN (SELECT districtid,namedistrict

                                        FROM district) dis

                                  ON dis.districtid = p.districtid

                                  JOIN (SELECT wardid,nameward

                                        FROM ward) w

                                  ON w.wardid = p.wardid

                                  JOIN street st

                                  ON st.streetid = p.streetid

                     where  p.reid = pr\_reid;

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

            SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

        ELSE

            SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                        SELECT p.reid, p.title, p.description, p.price

                            , p.area, p.viewCount, p.phone

                            , p.address, p.rented, p.createdat

                            , c.name as 'categories', pr.nameprovince

                            , dis.namedistrict, w.nameward,st.namestreet as 'street'

                            , pro.nameproject ,u.name, u.email, u.phone

                     FROM posts p JOIN users u

                                  ON u.userid = p.userid

                                  JOIN categories c

                                  ON c.categoryid = p.categoryid

                                  JOIN (SELECT provinceid,nameprovince

                                        FROM province ) pr

                                  ON pr.provinceid = p.provinceid

                                  JOIN (SELECT districtid,namedistrict

                                        FROM district) dis

                                  ON dis.districtid = p.districtid

                                  JOIN (SELECT wardid,nameward

                                        FROM ward) w

                                  ON w.wardid = p.wardid

                                  JOIN street st

                                  ON st.streetid = p.streetid

                                  JOIN (SELECT projectid,nameproject

                                        FROM project ) pro

                                  ON pro.projectid = p.projectid

                     where  p.reid = pr\_reid;

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

            SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

        END IF;

    END IF;

END; $$

use bds;

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_get\_convenient $$

CREATE PROCEDURE sp\_get\_convenient(pr\_reid char(40))

BEGIN

    DECLARE isExist int DEFAULT -1;

    SELECT COUNT(\*) INTO isExist

    FROM  convenient c

    WHERE c.reid = pr\_reid;

    IF (isExist = 0) THEN

        SELECT 'Không có tiện ghi nào cả !';

    ELSE

        SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

        START TRANSACTION;

            SET SQL\_SAFE\_UPDATES = 0;

                SELECT bedroom, bathroom, floor, direction

                      , balconyDirection, furniture, fontageArea

                from convenient c

                where c.reid = pr\_reid;

            SET SQL\_SAFE\_UPDATES = 1;

        COMMIT;

        SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

    END IF;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_get\_listMedias $$

CREATE PROCEDURE sp\_get\_listMedias(pr\_reid char(40))

BEGIN

    DECLARE isExist int DEFAULT -1;

    SELECT COUNT(\*) INTO isExist

    FROM medias m

    WHERE m.reid = pr\_reid;

    IF (isExist = 0) THEN

        SELECT 'Không có hình ảnh nào cả !';

    ELSE

        SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

        START TRANSACTION;

            SET SQL\_SAFE\_UPDATES = 0;

                SELECT url

                from medias

                where reid = pr\_reid;

            SET SQL\_SAFE\_UPDATES = 1;

        COMMIT;

        SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

    END IF;

END; $$

#### Tìm kiếm từ khóa

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_search\_keyword $$

CREATE PROCEDURE sp\_search\_keyword(pr\_keyword text)

BEGIN

        SET pr\_keyword = CONCAT('%',pr\_keyword,'%');

        SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                    SELECT  \*

                    FROM posts p JOIN medias m

                                ON m.reid = p.reid

                    WHERE p.title like pr\_keyword AND  p.approve = 0

                    group by p.reid;

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

        SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

END; $$

#### Tìm kiếm địa chỉ theo thành phố

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Province\_Posts $$

CREATE PROCEDURE sp\_Province\_Posts(pr\_province\_id INT)

BEGIN

    DECLARE isExitsProvince INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExitsProvince

    FROM posts p

    WHERE  p.provinceid = pr\_province\_id

           AND  p.approve = 0;

    IF isExitsProvince <= 0 THEN

        SELECT 0,"Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p JOIN medias m

                     ON m.reid = p.reid

        WHERE p.provinceid = pr\_province\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo thành phố, quận

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Province\_District\_Posts $$

CREATE PROCEDURE sp\_Province\_District\_Posts(pr\_province\_id INT,pr\_district\_id INT)

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.provinceid = pr\_province\_id

           AND p.districtid  = pr\_district\_id

           AND  p.approve = 0;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p JOIN medias m

                     ON m.reid = p.reid

        WHERE p.provinceid = pr\_province\_id AND p.districtid  = pr\_district\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo thành phố, quận, phường

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Province\_District\_Ward\_Posts $$

CREATE PROCEDURE sp\_Province\_District\_Ward\_Posts(pr\_province\_id INT,pr\_district\_id INT

                                                 ,pr\_ward\_id INT)

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.provinceid = pr\_province\_id AND p.districtid  = pr\_district\_id

           AND p.wardid = pr\_ward\_id AND  p.approve = 0;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p join medias m

                     on m.reid = p.reid

        WHERE  p.provinceid = pr\_province\_id AND p.districtid  = pr\_district\_id

               AND p.districtid = pr\_ward\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo dự án

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Project $$

CREATE PROCEDURE sp\_Project(pr\_project\_id INT)

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.projectid = pr\_project\_id AND  p.approve = 0;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p join medias m

                     on m.reid = p.reid

        WHERE  p.projectid = pr\_project\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo dự án và thành phố

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Project\_Province $$

CREATE PROCEDURE sp\_Project\_Province(pr\_project\_id int,pr\_province\_id int)

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.projectid = pr\_project\_id

           AND p.provinceid = pr\_province\_id

           AND  p.approve = 0;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p join medias m

                     on m.reid = p.reid

        WHERE  p.projectid = pr\_project\_id AND p.provinceid = pr\_province\_id

              AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo dự án, thành phố, quận

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Project\_Province\_District $$

CREATE PROCEDURE sp\_Project\_Province\_District(pr\_project\_id int,pr\_province\_id int

                                             ,pr\_district\_id int )

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.projectid = pr\_project\_id

         AND p.provinceid = pr\_province\_id AND p.districtid = pr\_district\_id

         AND  p.approve = 0;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p join medias m

                    on m.reid = p.reid

        WHERE  p.projectid = pr\_project\_id AND p.provinceid = pr\_province\_id

               AND p.districtid = pr\_district\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Tìm kiếm địa chỉ theo dự án, thành phố, quận, đường, phường

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Province\_District\_Ward\_Street\_Posts $$

CREATE PROCEDURE sp\_Province\_District\_Ward\_Street\_Posts(pr\_province\_id INT,pr\_district\_id INT

                                                 ,pr\_ward\_id int,pr\_street\_id INT)

BEGIN

    DECLARE isExists INT DEFAULT -1 ;

    SELECT COUNT(\*) into isExists

    FROM posts p

    WHERE  p.provinceid = pr\_province\_id AND p.districtid  = pr\_district\_id

           AND p.wardid = pr\_ward\_id AND p.streetid = pr\_street\_id

           AND  p.approve = 0 ;

    IF isExists <= 0 THEN

        SELECT "Hiện không có nhà đất nào tại đây";

    ELSE

        SELECT \*

        FROM posts p join medias m

                     on m.reid = p.reid

        WHERE  p.provinceid = pr\_province\_id AND p.districtid  = pr\_district\_id

               AND p.wardid = pr\_ward\_id AND p.streetid = pr\_street\_id AND  p.approve = 0

        GROUP BY p.reid;

    END IF ;

END; $$

#### Xem danh sách bài đăng

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_get\_post $$

CREATE PROCEDURE sp\_get\_post()

BEGIN

        START TRANSACTION;

                SELECT \*

                FROM posts p JOIN (SELECT reid , url

                       FROM medias) m

                ON p.reid = m.reid

                GROUP BY p.reid;

        COMMIT;

END; $$

#### Yêu thích bài đăng

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_savePosts $$

CREATE PROCEDURE sp\_savePosts(pr\_reid char(40), pr\_userid char(40))

BEGIN

        DECLARE isExists int DEFAULT -1;

        DECLARE savePost\_id char(40) DEFAULT uuid();

        DECLARE savePost\_id\_exists char(40) DEFAULT -1;

        SELECT COUNT(\*) INTO isExists

        FROM saveposts s

        WHERE s.reid = pr\_reid AND s.userid = pr\_userid;

        IF (isExists>0) THEN

            SELECT savePostId  into savePost\_id\_exists

            FROM saveposts s

            WHERE s.reid = pr\_reid AND s.userid = pr\_userid;

            SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                    DELETE FROM saveposts

                    WHERE saveposts.reid = pr\_reid AND saveposts.userid = pr\_userid;

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

            SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

            SELECT savePost\_id\_exists,0;

        ELSE

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                INSERT INTO saveposts(savePostId,reid,userid)

                VALUES (savePost\_id,pr\_reid,pr\_userid);

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

            SELECT \*

            FROM saveposts s

            WHERE s.reid = pr\_reid AND s.userid = pr\_userid ;

        END IF;

END; $$

#### Sữa thông tin tài khoản

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Edit\_User $$

CREATE PROCEDURE sp\_Edit\_User(pr\_userid char(40),pr\_name varchar(150), pr\_phone varchar(100))

BEGIN

    DECLARE isExistsUser INT DEFAULT -1;

    SELECT COUNT(\*) INTO isExistsUser

    FROM users u

    WHERE u.userid = pr\_userid;

    IF isExistsUser <= 0 THEN

        SELECT "USER KHÔNG TỒN TẠI";

    ELSE

        START TRANSACTION;

            SET SQL\_SAFE\_UPDATES=0;

            UPDATE users

            SET name = pr\_name , phone = pr\_phone

            WHERE  userid = pr\_userid;

            SET SQL\_SAFE\_UPDATES=1;

        COMMIT;

        SELECT userid,name,username,email,phone

        FROM users u

        WHERE u.userid = pr\_userid;

    END IF;

END; $$

#### Bình luận

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_Comment $$

CREATE PROCEDURE sp\_Comment(pr\_reid char(40), pr\_userid char(40),pr\_content text)

BEGIN

    DECLARE isExistsRe int DEFAULT -1;

    DECLARE isExistsUser int DEFAULT -1;

    DECLARE contentLen int DEFAULT -1;

    DECLARE isExistsComment int DEFAULT -1;

    DECLARE comment\_id char(40) DEFAULT uuid();

    SELECT COUNT(\*) INTO isExistsRe

    FROM posts p

    WHERE p.reid = pr\_reid;

    SELECT COUNT(\*) INTO isExistsUser

    FROM  users u

    WHERE u.userid = pr\_userid;

    SELECT LENGTH(pr\_content) INTO contentLen;

    SELECT COUNT(\*) INTO isExistsComment

    FROM comment c

    WHERE c.reid = pr\_reid AND c.userid = pr\_userid;

    IF (isExistsRe <= 0) THEN

            SELECT 'Nhà đất không có để comment !' , 0;

        ELSEIF (isExistsUser <= 0)  THEN

            SELECT 'Người dùng không tồn tại trong hệ thống!' , 0;

        ELSEIF (contentLen < 10) THEN

            SELECT 'Nội dung comment không phù  hợp' , 0;

        ELSEIF (isExistsComment > 0) THEN

            SELECT 'Bạn đã comment tại bài đăng này', 0;

        ELSE

            START TRANSACTION;

                SET SQL\_SAFE\_UPDATES = 0;

                    INSERT INTO comment(commentid,content,reid,userid)

                    VALUES (comment\_id,pr\_content,pr\_reid,pr\_userid);

                SET SQL\_SAFE\_UPDATES = 1;

            COMMIT;

            SELECT c.content, u.name , u.userid , 1

            FROM comment c

            JOIN users u ON c.userid = u.userid

            WHERE c.commentid = comment\_id;

    END IF;

END; $$

#### Xem danh sách chưa chặn bài đăng

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_return\_list\_posts $$

CREATE PROCEDURE sp\_return\_list\_posts(pr\_id\_user varchar(50))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    SET is\_admin = fnc\_checkAuthorization(pr\_id\_user);

    IF is\_admin = 1 THEN

        SELECT \*

        FROM posts p JOIN (

                            SELECT reid,url

                            FROM medias

                            ) m

                     ON p.reid = m.reid

        WHERE p.approve = 0

        GROUP BY p.reid;

    ELSE

        SELECT 0 as "status","Not right admin" as "message",'' as "data";

    END IF;

END; $$

#### Chặn bài đăng

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_is\_exists\_posts$$

CREATE FUNCTION fnc\_is\_exists\_posts(pr\_posts char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_exists char(40) DEFAULT NULL;

     SELECT COUNT(reid) INTO is\_exists

     FROM posts p

     WHERE p.reid = pr\_posts;

     IF is\_exists > 0 THEN

        RETURN TRUE;

     END IF;

     RETURN FALSE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_blocked$$

CREATE FUNCTION fnc\_check\_blocked(pr\_posts char(40))

RETURNS boolean

DETERMINISTIC

begin

    DECLARE check\_blocked INT DEFAULT -1;

    SELECT COUNT(reid) INTO check\_blocked

    FROM posts p

    WHERE  p.reid = pr\_posts AND p.approve = 1;

    IF check\_blocked > 0 THEN

        RETURN TRUE;

    END IF;

    RETURN FALSE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_bans\_post $$

CREATE PROCEDURE sp\_bans\_post(pr\_idUser char(40), pr\_id\_post char(40))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    DECLARE is\_exists\_posts INT DEFAULT -1;

    DECLARE is\_check\_blocked INT DEFAULT -1;

    SET is\_exists\_posts = fnc\_is\_exists\_posts(pr\_id\_post);

    IF is\_exists\_posts = 0 THEN

        SELECT "Post not exists";

    ELSE

        SET is\_admin = fnc\_check\_authorization(pr\_idUser);

        IF is\_admin = 1 THEN

            SET is\_check\_blocked = fnc\_check\_blocked(pr\_id\_post);

            IF is\_check\_blocked > 0 THEN

                SELECT 0 as "status", "POST BLOCKED" as "message",'' as "data";

            ELSE

                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

                    START TRANSACTION;

                        SET SQL\_SAFE\_UPDATES = 0;

                            UPDATE Posts

                            SET approve = 1

                            WHERE reid = pr\_id\_post;

                        SET SQL\_SAFE\_UPDATES = 1;

                    COMMIT;

                    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

                    SELECT 1 as "status","SUCCESS UPDATE" as "message",'' as "data"  ;

            END IF;

        ELSE

            SELECT 0 as "status","Not right admin" as "message",'' as "data";

        END IF;

    END IF;

END; $$

#### Xem danh sách bài đăng đã bị chặn

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_return\_list\_posts\_blocked $$

CREATE PROCEDURE sp\_return\_list\_posts\_blocked(pr\_id\_user varchar(50))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    SET is\_admin = fnc\_checkAuthorization(pr\_id\_user);

    IF is\_admin = 1 THEN

        SELECT \*

        FROM posts p JOIN (

                            SELECT reid,url

                            FROM medias

                            ) m

                     ON p.reid = m.reid

        WHERE p.approve = 1

        GROUP BY p.reid;

    ELSE

        SELECT 0 as "status","Not right admin" as "message",'' as "data";

    END IF;

END; $$

#### Hủy chặn bài đăng

DROP FUNCTION IF EXISTS fnc\_is\_exists\_posts$$

CREATE FUNCTION fnc\_is\_exists\_posts(pr\_posts char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_exists char(40) DEFAULT NULL;

     SELECT COUNT(reid) INTO is\_exists

     FROM posts p

     WHERE p.reid = pr\_posts;

     IF is\_exists > 0 THEN

        RETURN TRUE;

     END IF;

     RETURN FALSE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_blocked$$

CREATE FUNCTION fnc\_check\_blocked(pr\_posts char(40))

RETURNS boolean

DETERMINISTIC

begin

    DECLARE check\_blocked INT DEFAULT -1;

    SELECT COUNT(reid) INTO check\_blocked

    FROM posts p

    WHERE  p.reid = pr\_posts AND p.approve = 1;

    IF check\_blocked > 0 THEN

        RETURN TRUE;

    END IF;

    RETURN FALSE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_recovery\_post $$

CREATE PROCEDURE sp\_recovery\_post(pr\_id\_user char(40), pr\_id\_post char(40))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    DECLARE is\_exists\_posts INT DEFAULT -1;

    DECLARE is\_check\_blocked INT DEFAULT -1;

    SET is\_exists\_posts = fnc\_is\_exists\_posts(pr\_id\_post);

    IF is\_exists\_posts = 0 THEN

        SELECT 0 as "status","Post not exists" as "message",'' as 'data';

    ELSE

        SET is\_admin = fnc\_check\_authorization(pr\_id\_user);

        IF is\_admin = 1 THEN

            SET is\_check\_blocked = fnc\_check\_blocked(pr\_id\_post);

            IF is\_check\_blocked = 0 THEN

                SELECT 0 as "status", "POST NOT BLOCKED" as "message",'' as "data";

            ELSE

                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

                    START TRANSACTION;

                        SET SQL\_SAFE\_UPDATES = 0;

                            UPDATE Posts

                            SET approve = 0

                            WHERE reid = pr\_id\_post;

                        SET SQL\_SAFE\_UPDATES = 0;

                    COMMIT;

                    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

                    SELECT 1 as "status","SUCCESS UPDATE" as "message",'' as "data"  ;

            END IF;

        ELSE

            SELECT 0 as "status","Not right admin" as "message",'' as "data";

        END IF;

    END IF;

END; $$

#### Xem danh sách chưa chặn tài khoản

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_return\_list\_users $$

CREATE PROCEDURE sp\_return\_list\_users(pr\_id\_user varchar(50))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    SET is\_admin = fnc\_checkAuthorization(pr\_id\_user);

    IF is\_admin = 1 THEN

        SELECT \*

        FROM users u

        WHERE u.status = 0;

    ELSE

        SELECT 0 as "status","Bạn Không Phải Admin" as "message",'' as "data";

    END IF;

END; $$

#### Chặn tài khoản

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_is\_exists\_user$$

CREATE FUNCTION fnc\_is\_exists\_user(pr\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_exists char(40) DEFAULT NULL;

     SELECT COUNT(\*) INTO is\_exists

     FROM users u

     WHERE u.userid = pr\_user;

     IF is\_exists > 0 THEN

        RETURN TRUE;

     END IF;

     RETURN FALSE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_blocked\_user$$

CREATE FUNCTION fnc\_check\_blocked\_user(pr\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

    DECLARE check\_blocked INT DEFAULT -1;

    SELECT COUNT(\*) INTO check\_blocked

    FROM users u

    WHERE  u.userid = pr\_user AND  u.status = 1;

    IF check\_blocked > 0 THEN

        RETURN TRUE;

    END IF;

    RETURN FALSE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_block\_user $$

CREATE PROCEDURE sp\_block\_user(pr\_id\_user char(40), pr\_id\_user\_block char(40))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    DECLARE is\_exists\_user INT DEFAULT -1;

    DECLARE is\_check\_blocked INT DEFAULT -1;

    SET is\_exists\_user = fnc\_is\_exists\_user(pr\_id\_user\_block);

    IF is\_exists\_user = 0 THEN

        SELECT 0 as "status", "User không tồn tại" as "message", "" as "data";

    ELSE

        SET is\_admin = fnc\_check\_authorization(pr\_id\_user);

        IF is\_admin = 1 THEN

            SET is\_check\_blocked = fnc\_check\_blocked\_user(pr\_id\_user\_block);

            IF is\_check\_blocked > 0 THEN

                SELECT 0 as "status", "User đã được block" as "message",'' as "data";

            ELSE

                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

                    START TRANSACTION;

                        SET SQL\_SAFE\_UPDATES = 0;

                            UPDATE users

                            SET status = 1

                            WHERE userid = pr\_id\_user\_block;

                        SET SQL\_SAFE\_UPDATES = 1;

                    COMMIT;

                    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

                    SELECT 1 as "status","Update thành công" as "message",'' as "data"  ;

            END IF;

        ELSE

            SELECT 0 as "status","Bạn Không Phải Admin" as "message",'' as "data";

        END IF;

    END IF;

END; $$

#### Xem danh sách tài khoản đã bị chặn

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_return\_list\_users $$

CREATE PROCEDURE sp\_return\_list\_users(pr\_id\_user varchar(50))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    SET is\_admin = fnc\_checkAuthorization(pr\_id\_user);

    IF is\_admin = 1 THEN

        SELECT \*

        FROM users u

        WHERE u.status = 0;

    ELSE

        SELECT 0 as "status","Bạn Không Phải Admin" as "message",'' as "data";

    END IF;

END; $$

#### Hủy chặn tài khoản

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_authorization$$

CREATE FUNCTION fnc\_check\_authorization(pr\_id\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_admin char(40) DEFAULT NULL;

     SELECT u.name into is\_admin

     FROM users u

     WHERE u.typeuser = 1 AND  u.userid = pr\_id\_user;

     IF is\_admin IS NULL THEN

        RETURN FALSE;

     END IF;

    RETURN TRUE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_is\_exists\_user$$

CREATE FUNCTION fnc\_is\_exists\_user(pr\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

     DECLARE is\_exists char(40) DEFAULT NULL;

     SELECT COUNT(\*) INTO is\_exists

     FROM users u

     WHERE u.userid = pr\_user;

     IF is\_exists > 0 THEN

        RETURN TRUE;

     END IF;

     RETURN FALSE;

END; $$

DELIMITER $$

DROP FUNCTION IF EXISTS fnc\_check\_blocked\_user$$

CREATE FUNCTION fnc\_check\_blocked\_user(pr\_user char(40))

RETURNS boolean

DETERMINISTIC

begin

    DECLARE check\_blocked INT DEFAULT -1;

    SELECT COUNT(\*) INTO check\_blocked

    FROM users u

    WHERE  u.userid = pr\_user AND  u.status = 1;

    IF check\_blocked > 0 THEN

        RETURN TRUE;

    END IF;

    RETURN FALSE;

END; $$

DELIMITER $$

DROP PROCEDURE IF EXISTS sp\_block\_user $$

CREATE PROCEDURE sp\_block\_user(pr\_id\_user char(40), pr\_id\_user\_block char(40))

BEGIN

    DECLARE is\_admin INT DEFAULT 0;

    DECLARE is\_exists\_user INT DEFAULT -1;

    DECLARE is\_check\_blocked INT DEFAULT -1;

    SET is\_exists\_user = fnc\_is\_exists\_user(pr\_id\_user\_block);

    IF is\_exists\_user = 0 THEN

        SELECT 0 as "status", "User không tồn tại" as "message", "" as "data";

    ELSE

        SET is\_admin = fnc\_check\_authorization(pr\_id\_user);

        IF is\_admin = 1 THEN

            SET is\_check\_blocked = fnc\_check\_blocked\_user(pr\_id\_user\_block);

            IF is\_check\_blocked > 0 THEN

                SELECT 0 as "status", "User đã được block" as "message",'' as "data";

            ELSE

                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

                    START TRANSACTION;

                        SET SQL\_SAFE\_UPDATES = 0;

                            UPDATE users

                            SET status = 1

                            WHERE userid = pr\_id\_user\_block;

                        SET SQL\_SAFE\_UPDATES = 1;

                    COMMIT;

                    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

                    SELECT 1 as "status","Update thành công" as "message",'' as "data"  ;

            END IF;

        ELSE

            SELECT 0 as "status","Bạn Không Phải Admin" as "message",'' as "data";

        END IF;

    END IF;

END; $$

### Phân quyền

#### Tạo login mặc định

IF EXISTS(SELECT \* FROM master.dbo.syslogins WHERE NAME = 'TSK\_Client')

DROP LOGIN TSK\_Client

CREATE LOGIN TSK\_Client WITH PASSWORD = 'TOEIC\_SaoKhue'

GO

CREATE USER Client FOR LOGIN TSK\_Client

GO

#### Tạo Roles

CREATE ROLE IF NOT EXISTS 'RE\_QuanLy';

CREATE ROLE IF NOT EXISTS 'RE\_KhachHangThanThiet';

CREATE ROLE IF NOT EXISTS 'RE\_KhachHangVangLai';

#### Cấp quyền cho các Roles

GRANT SELECT, INSERT, UPDATE, DELETE ON bds.categories TO 'RE\_QuanLy';

GRANT INSERT ON bds.comment TO 'RE\_QuanLy';

GRANT INSERT ON bds.convenient TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.district TO 'RE\_QuanLy';

GRANT SELECT ON bds.medias TO 'RE\_QuanLy';

GRANT SELECT, UPDATE ON bds.posts TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.project TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.province TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.report TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.saveposts TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.street TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.users TO 'RE\_QuanLy';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.ward TO 'RE\_QuanLy';

GRANT SELECT ON bds.categories TO 'RE\_KhachHangThanThiet';

GRANT INSERT, UPDATE ON bds.comment TO 'RE\_KhachHangThanThiet';

GRANT INSERT, UPDATE ON bds.convenient TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.district TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.medias TO 'RE\_KhachHangThanThiet';

GRANT SELECT,INSERT, UPDATE, DELETE ON bds.posts TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.project TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.province TO 'RE\_KhachHangThanThiet';

GRANT SELECT,INSERT ON bds.report TO 'RE\_KhachHangThanThiet';

GRANT SELECT,INSERT, DELETE ON bds.saveposts TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.street TO 'RE\_KhachHangThanThiet';

GRANT SELECT,INSERT, UPDATE ON bds.users TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.ward TO 'RE\_KhachHangThanThiet';

GRANT SELECT ON bds.\* TO 'RE\_KhachHangVangLai';

GRANT EXECUTE ON PROCEDURE bds.sp\_Report TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Register TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_update\_status\_post TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_get\_listMedias TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_show\_detail\_info TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_get\_convenient TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_search\_keyword TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Province\_Posts TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Province\_District\_Posts TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Province\_District\_Ward\_Posts TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Project TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Project\_Province TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Project\_Province\_District TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Province\_District\_Ward\_Street\_Posts TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_get\_post TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_savePosts TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Edit\_User TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_Comment TO 'RE\_KhachHangThanThiet';

GRANT EXECUTE ON PROCEDURE bds.sp\_return\_list\_posts TO 'RE\_QuanLy';

GRANT EXECUTE ON PROCEDURE bds.sp\_bans\_post TO 'RE\_QuanLy';

GRANT EXECUTE ON PROCEDURE bds.sp\_return\_list\_posts\_blocked TO 'RE\_QuanLy';

GRANT EXECUTE ON PROCEDURE bds.sp\_recovery\_post TO 'RE\_QuanLy';

GRANT EXECUTE ON PROCEDURE bds.sp\_block\_user TO 'RE\_QuanLy';

GRANT EXECUTE ON PROCEDURE bds.sp\_return\_list\_users TO 'RE\_QuanLy';

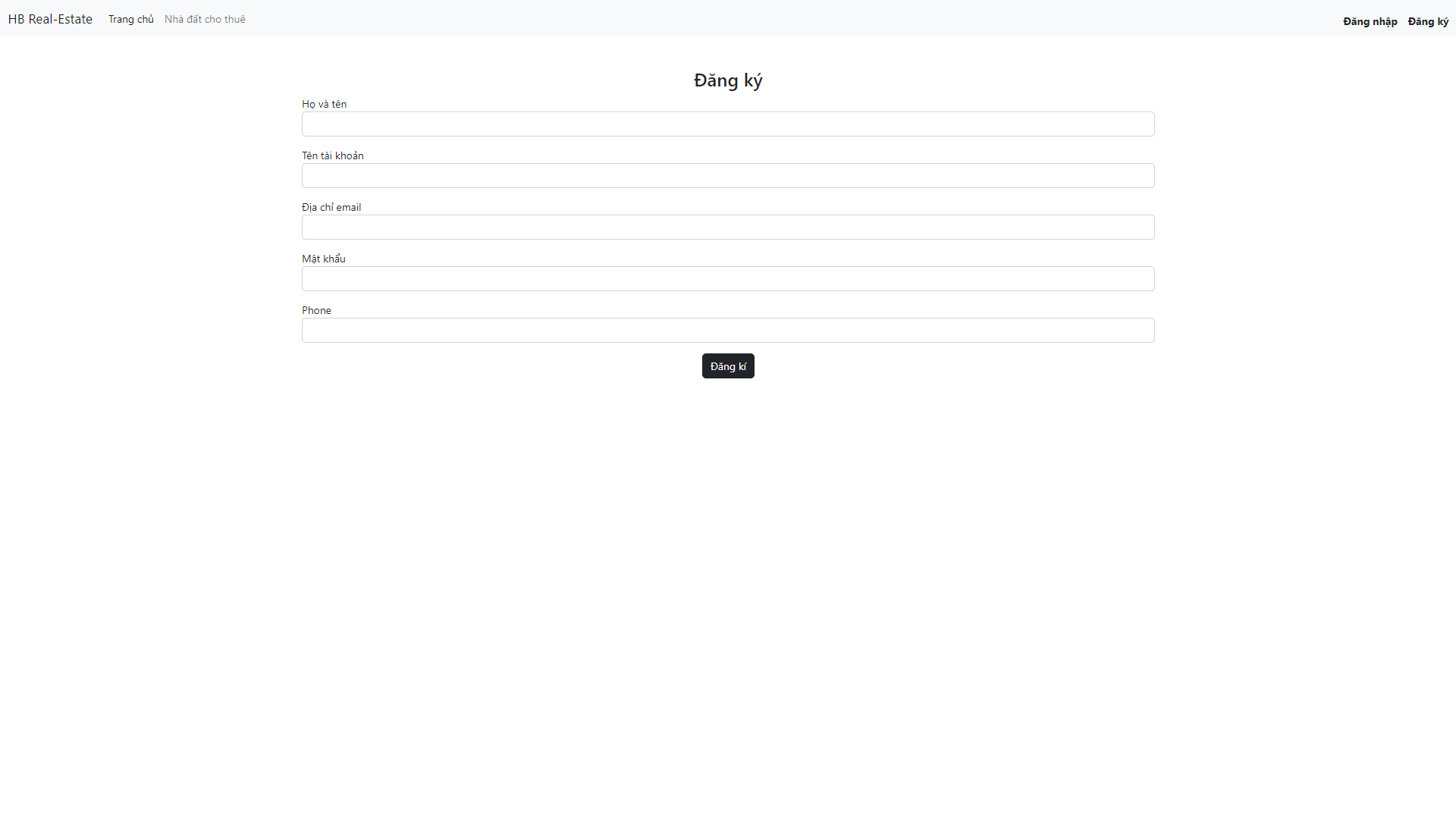
GRANT EXECUTE ON PROCEDURE bds.sp\_recovery\_user TO 'RE\_QuanLy';

## Phía Client

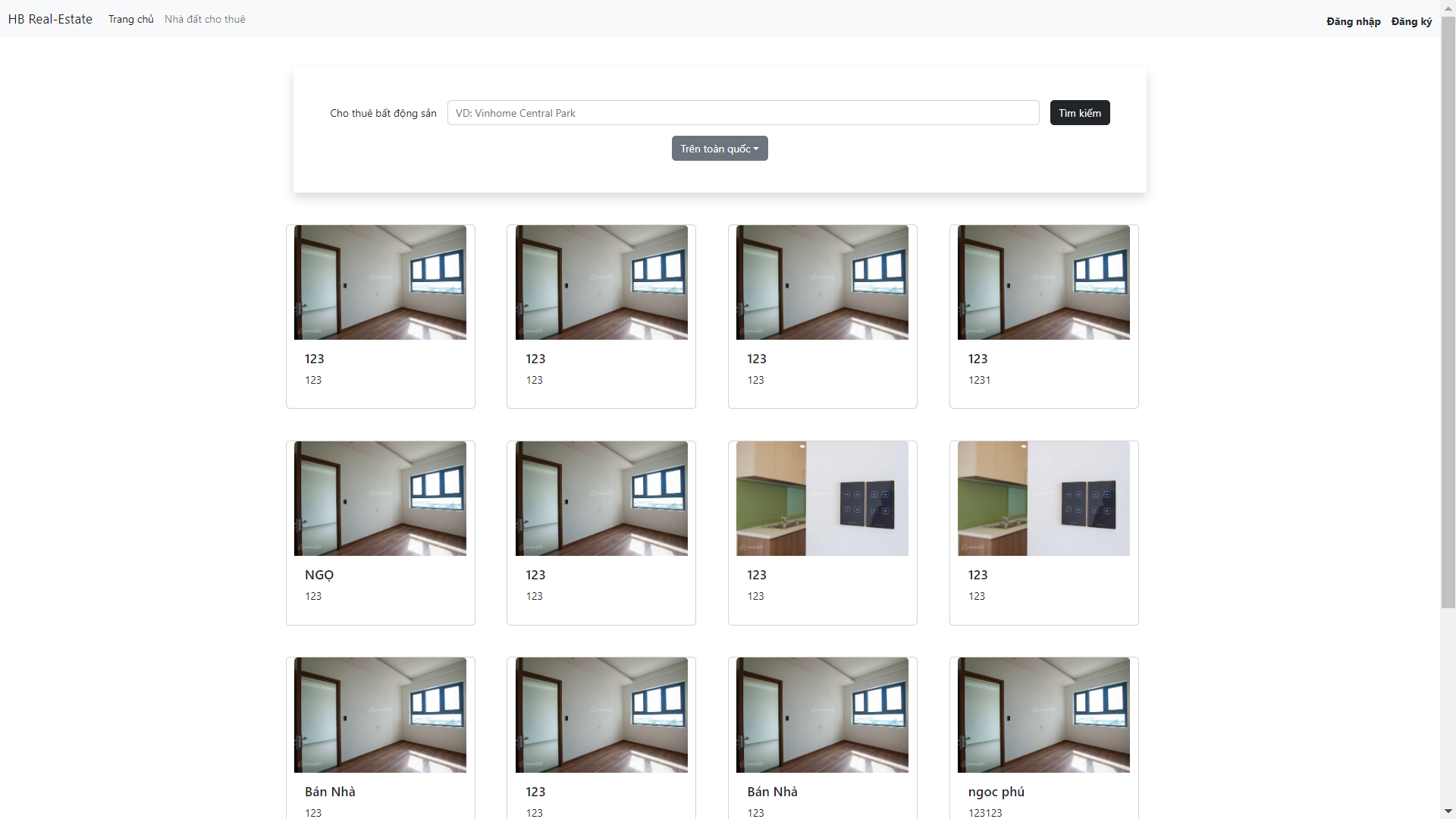
### Giao diện thực hiện các chức năng



Hình 13 Giao diện Đăng nhập

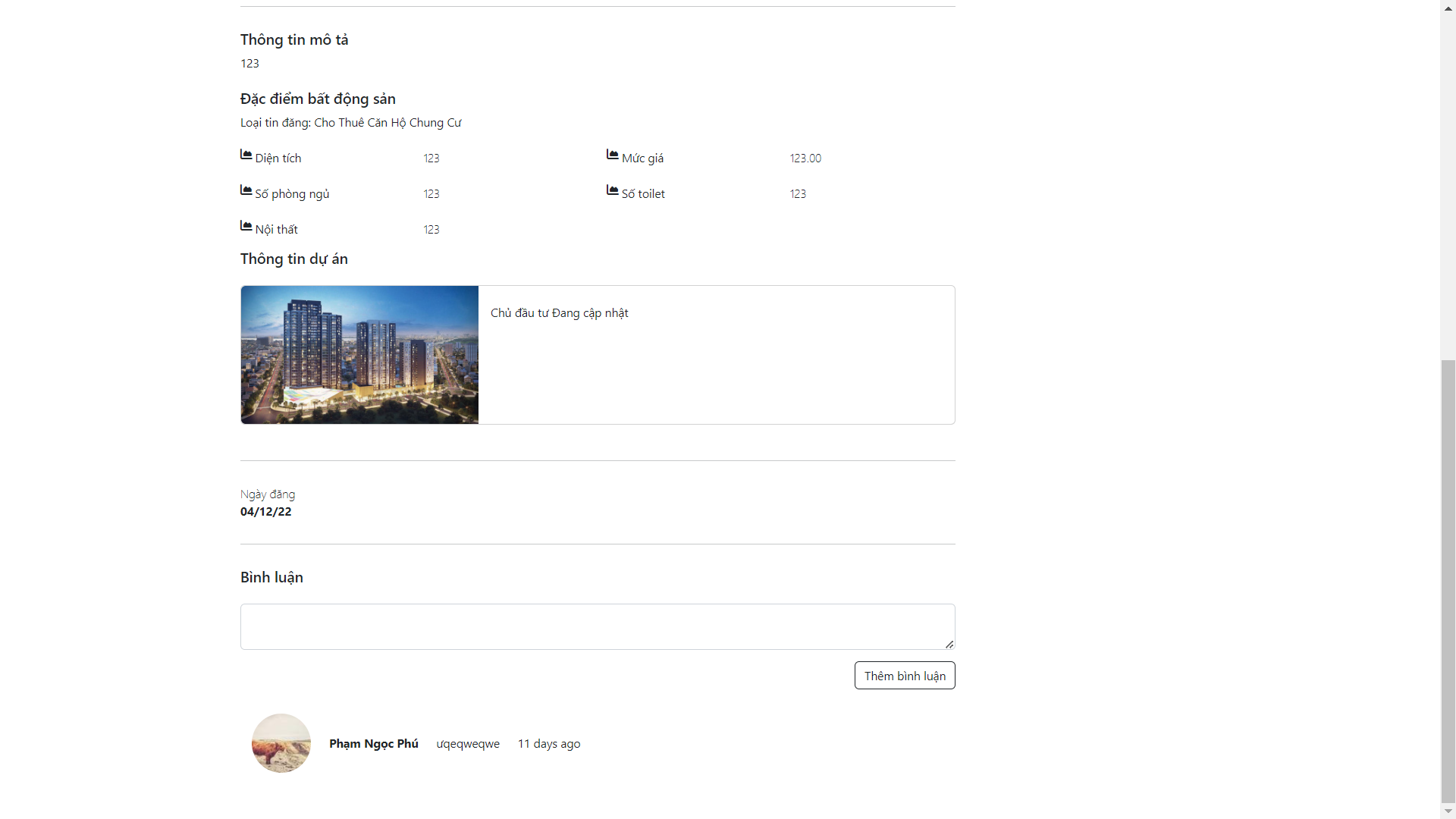


Hình 14 Giao diện Đăng ký

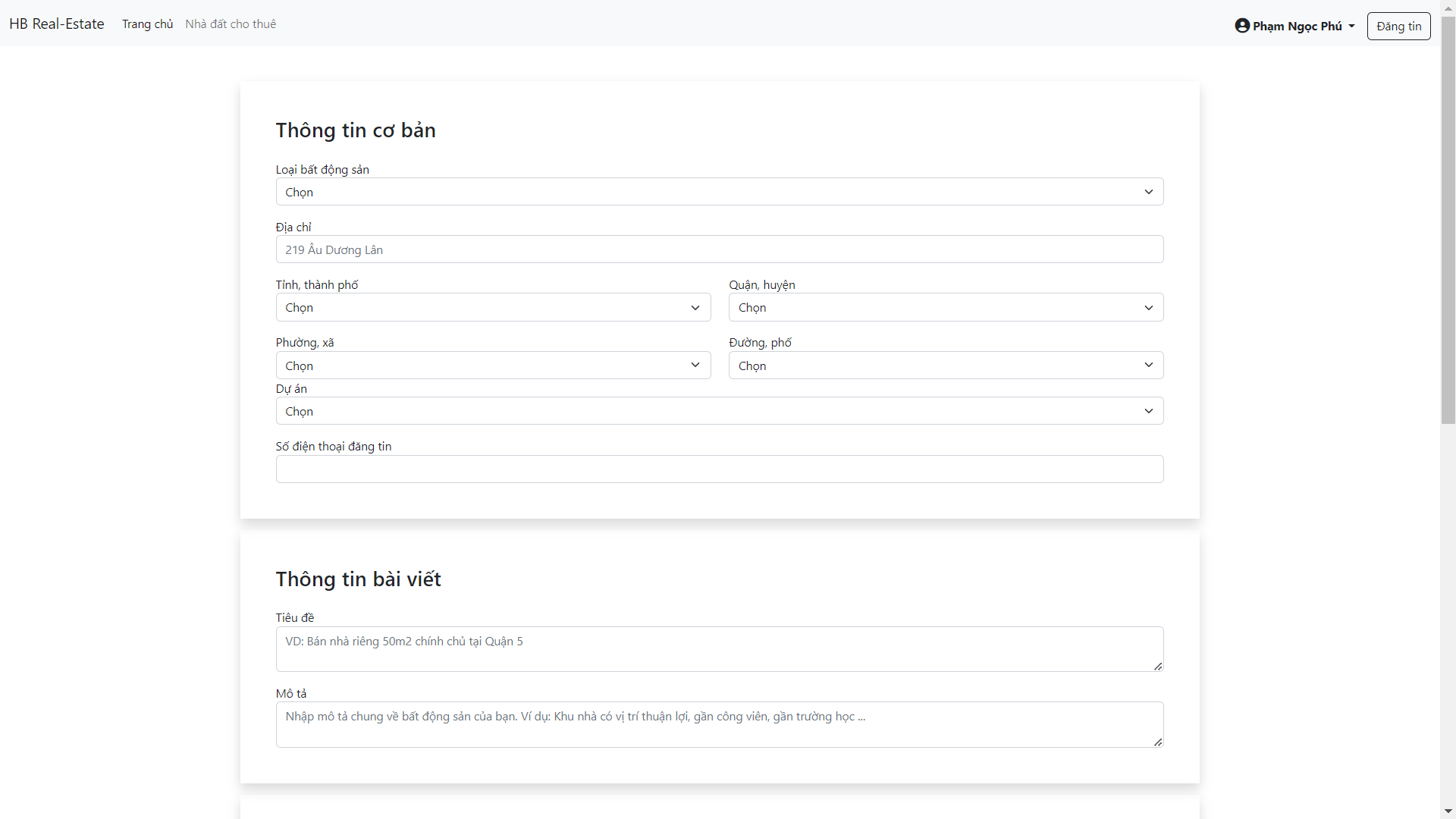


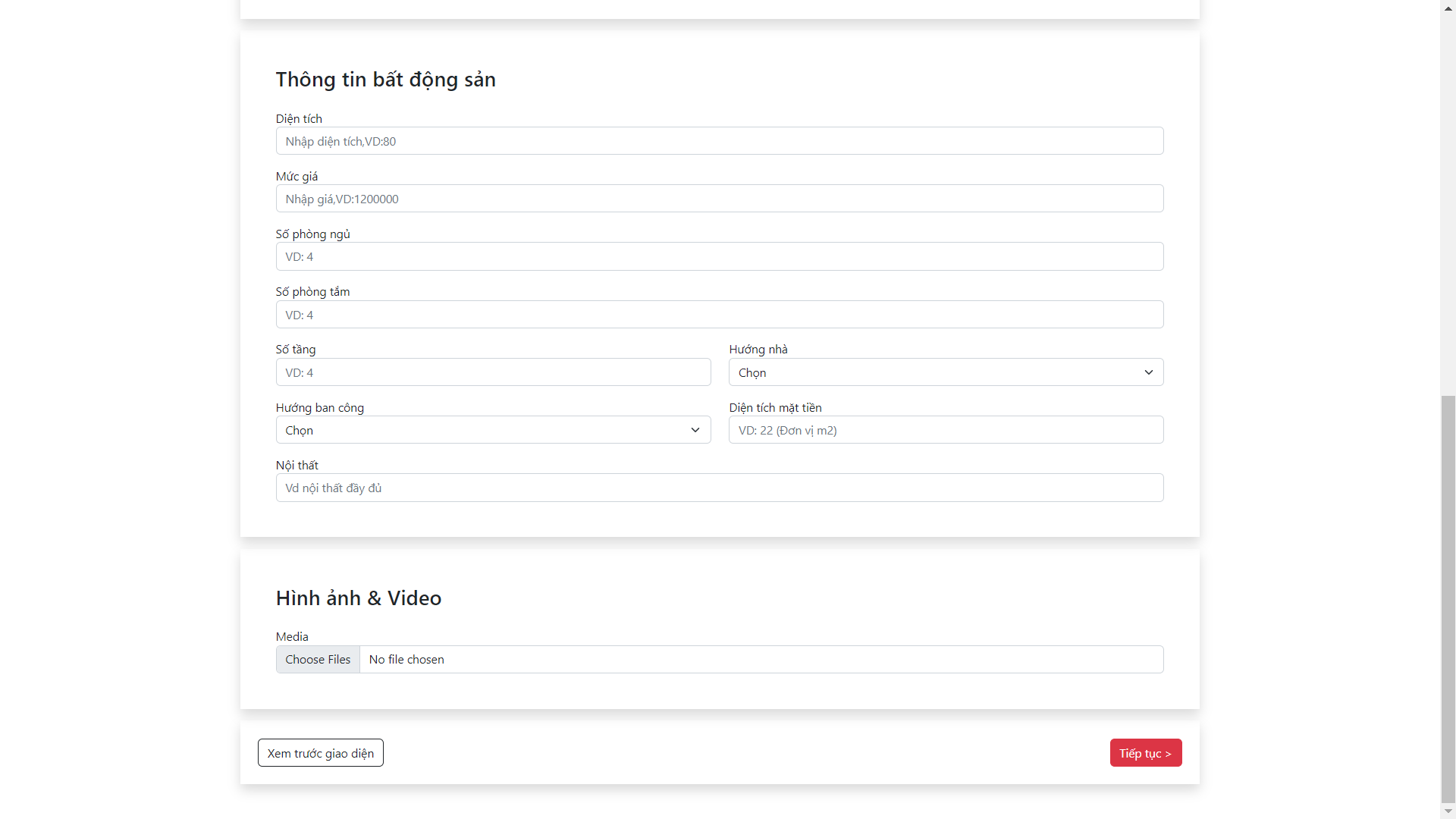
Hình 15 Giao diện Trang chủ



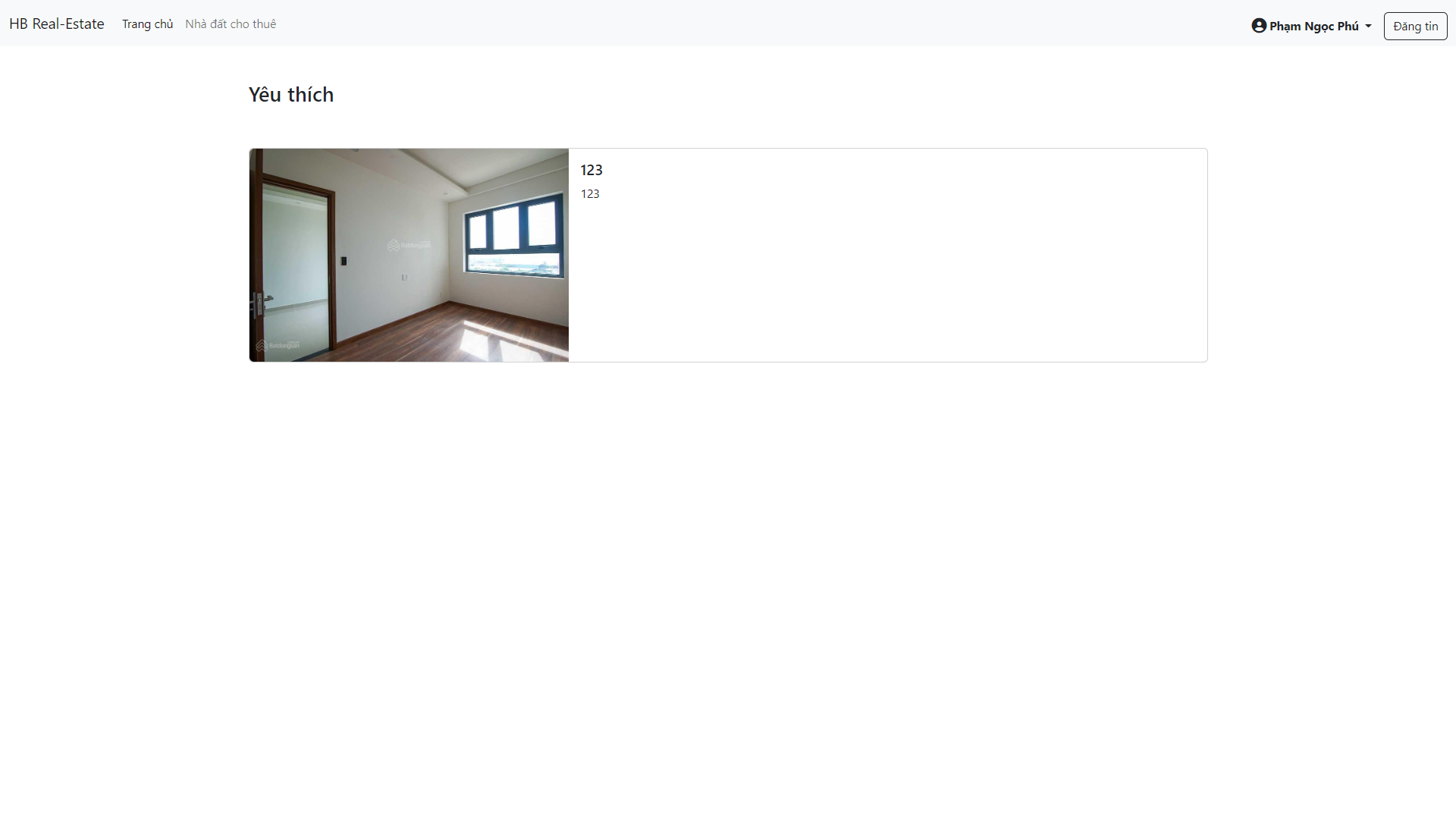


Hình 16 Giao diện chi tiết bài đăng

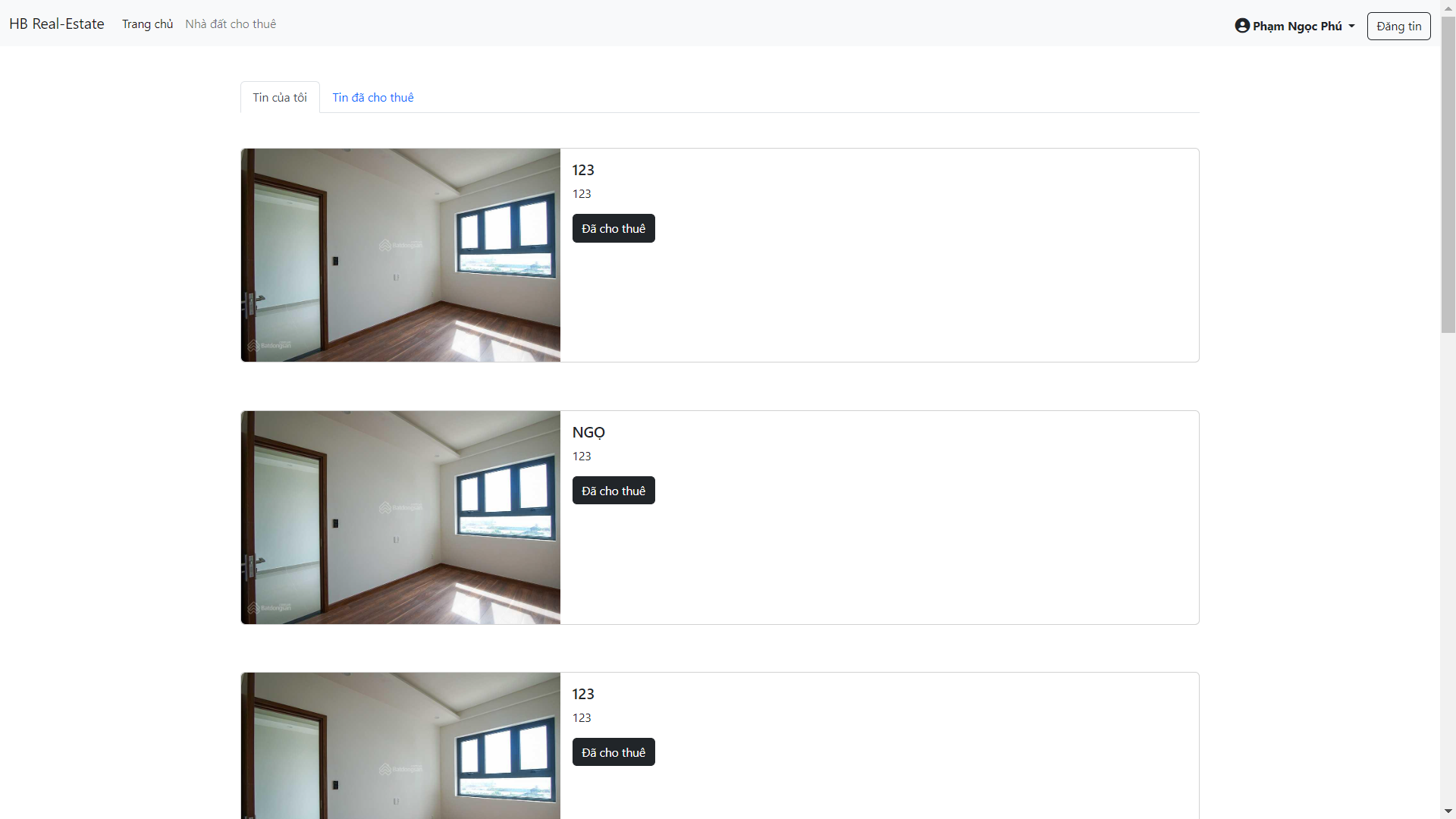




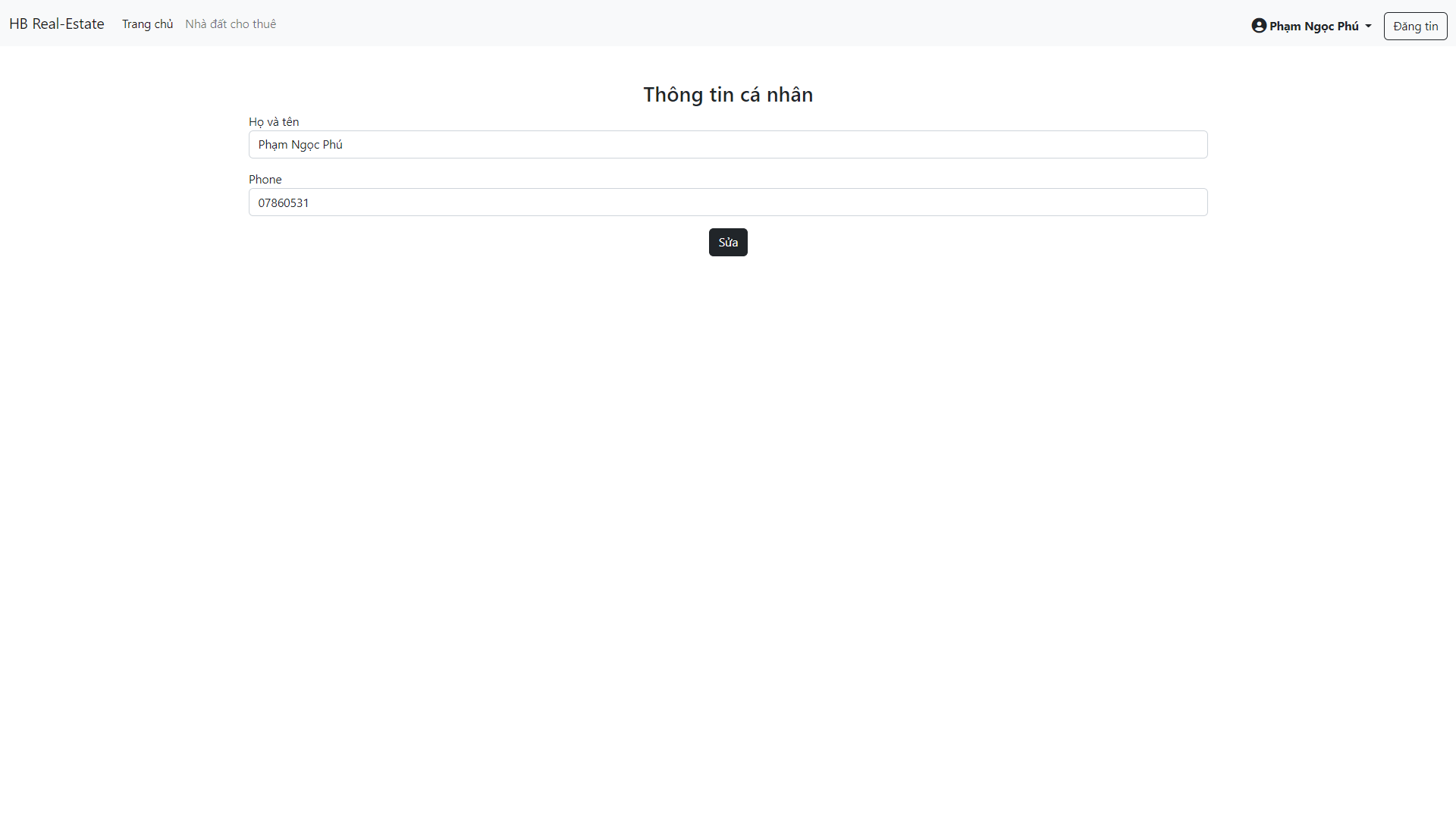
Hình 17 Giao diện đăng tin



Hình 20 Giao diện bài đăng đã yêu thích



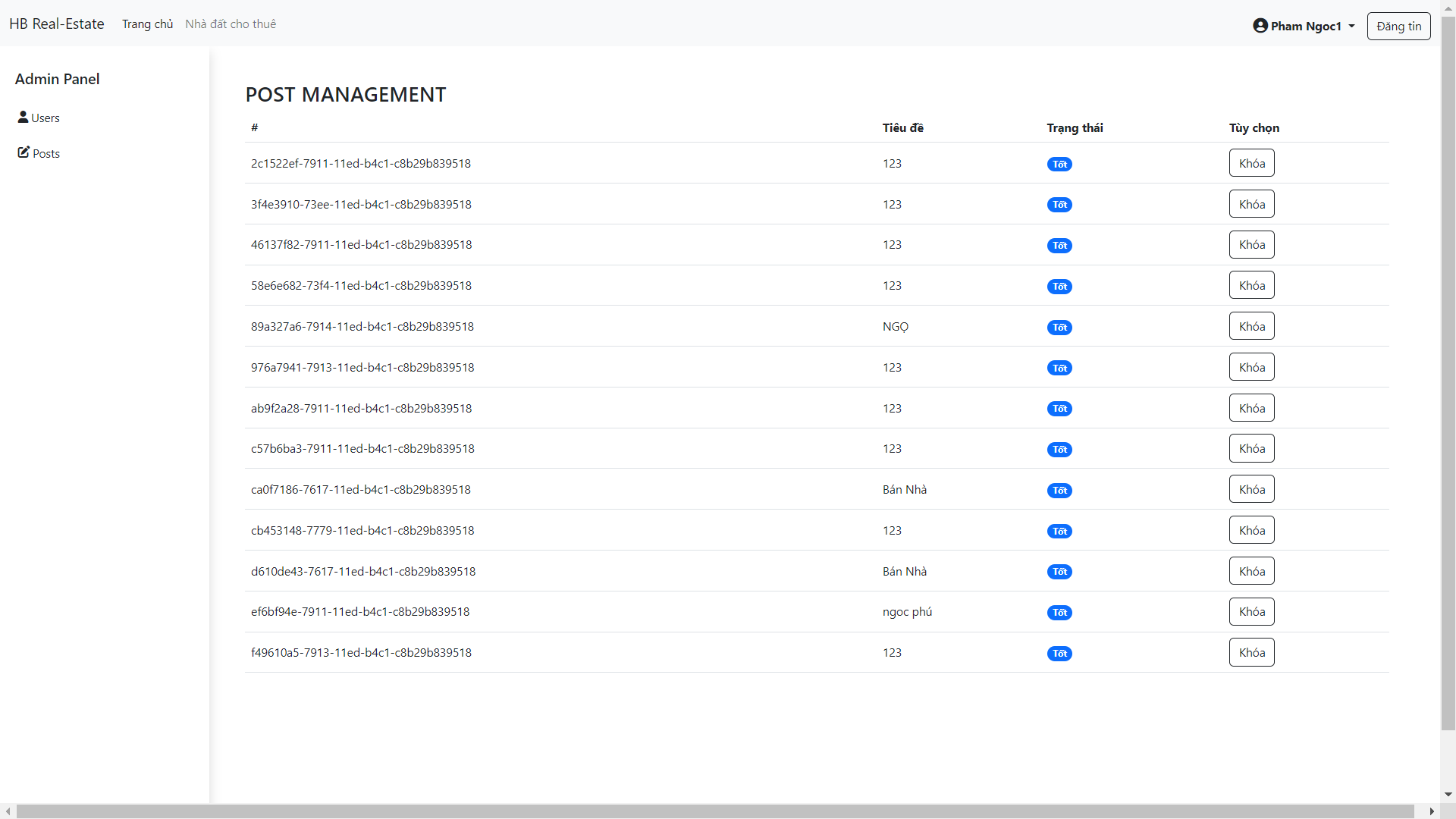
Hình 21 Giao diện bài đăng đã tạo



Hình 22 Giao diện sữa thông tin



Hình 23 Giao diện quản lý tài khoản

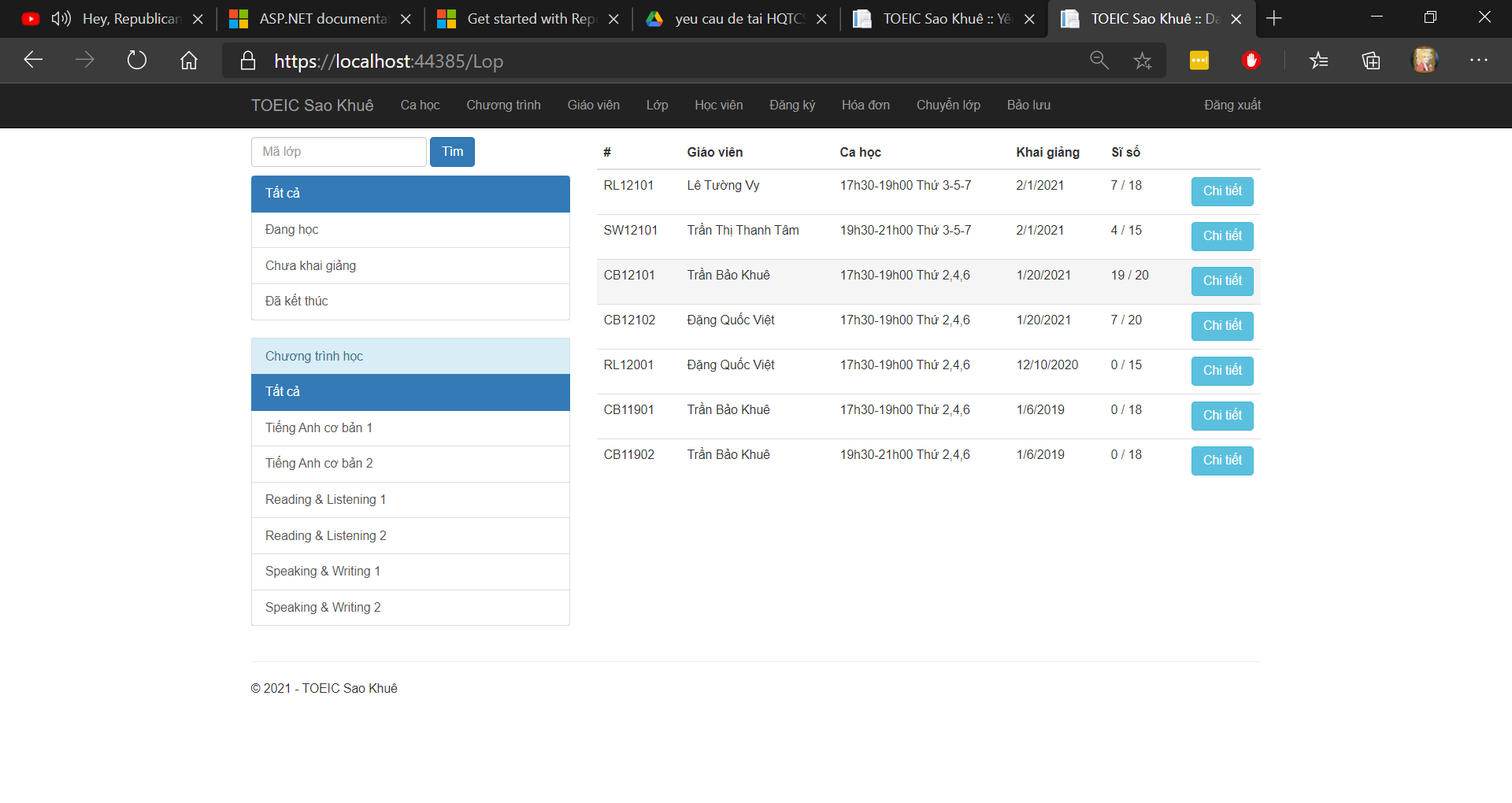


Hình 24 Giao diện quản lý bài đăng

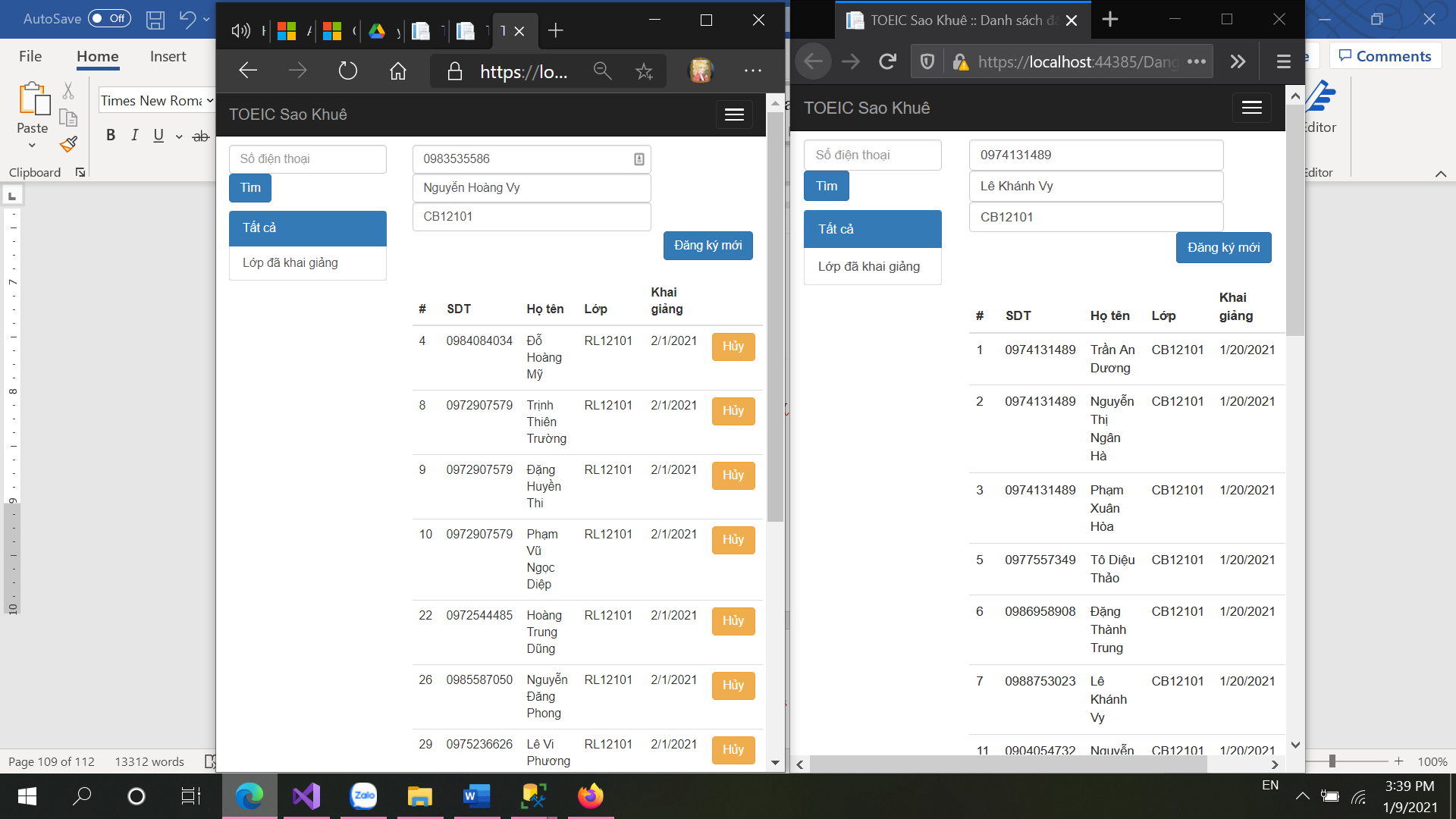
### Giả lập xử lý đồng thời

#### Đăng ký

Giả sử hai nhân viên tư vấn đồng thời đăng ký cho hai học viên khác nhau một lớp chỉ còn một suất.

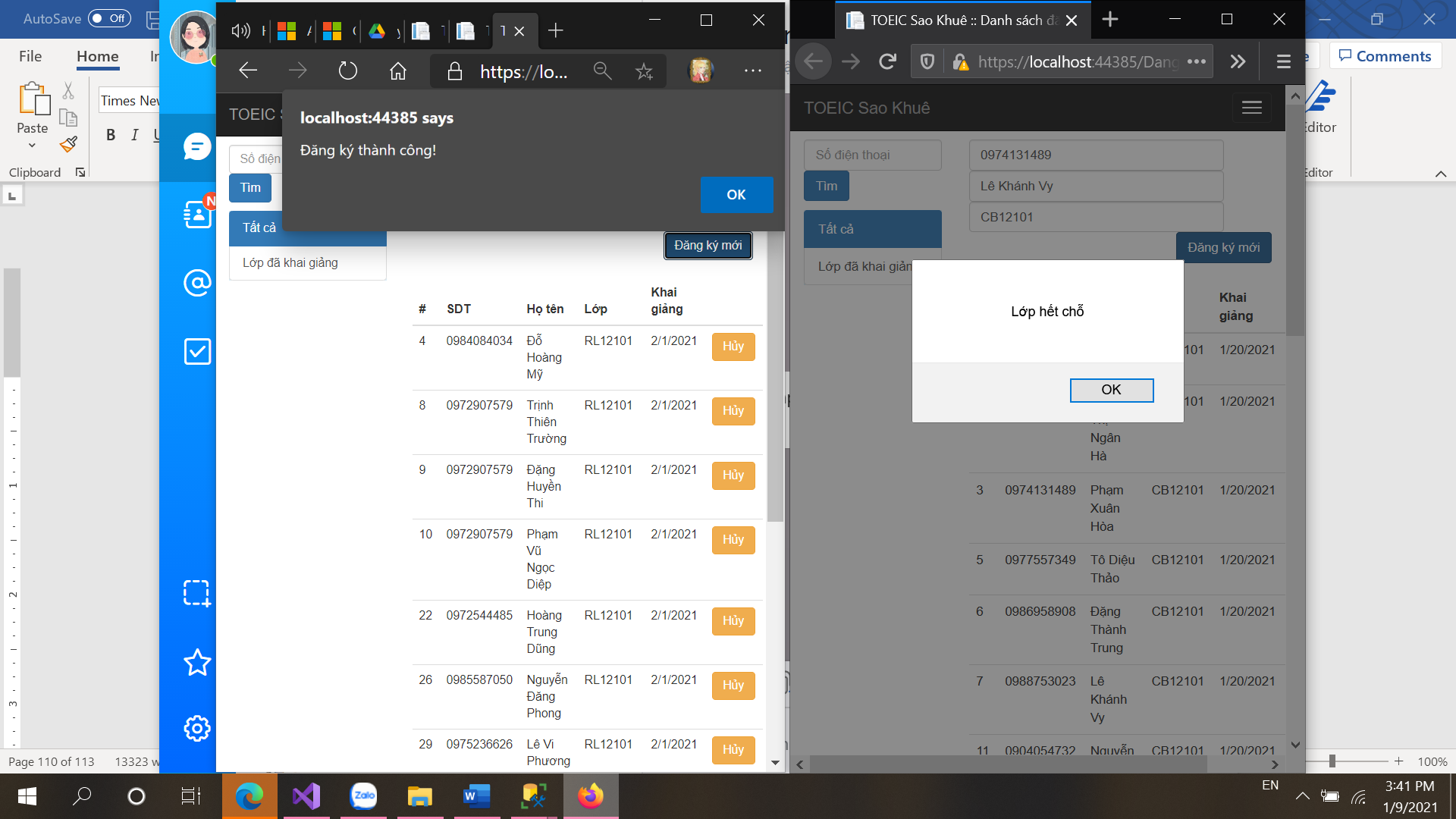


Hình 25 Giả lập xử lý đồng thời – Đăng ký (1)



Hình 26 Giả lập xử lý đồng thời – Đăng ký (2)

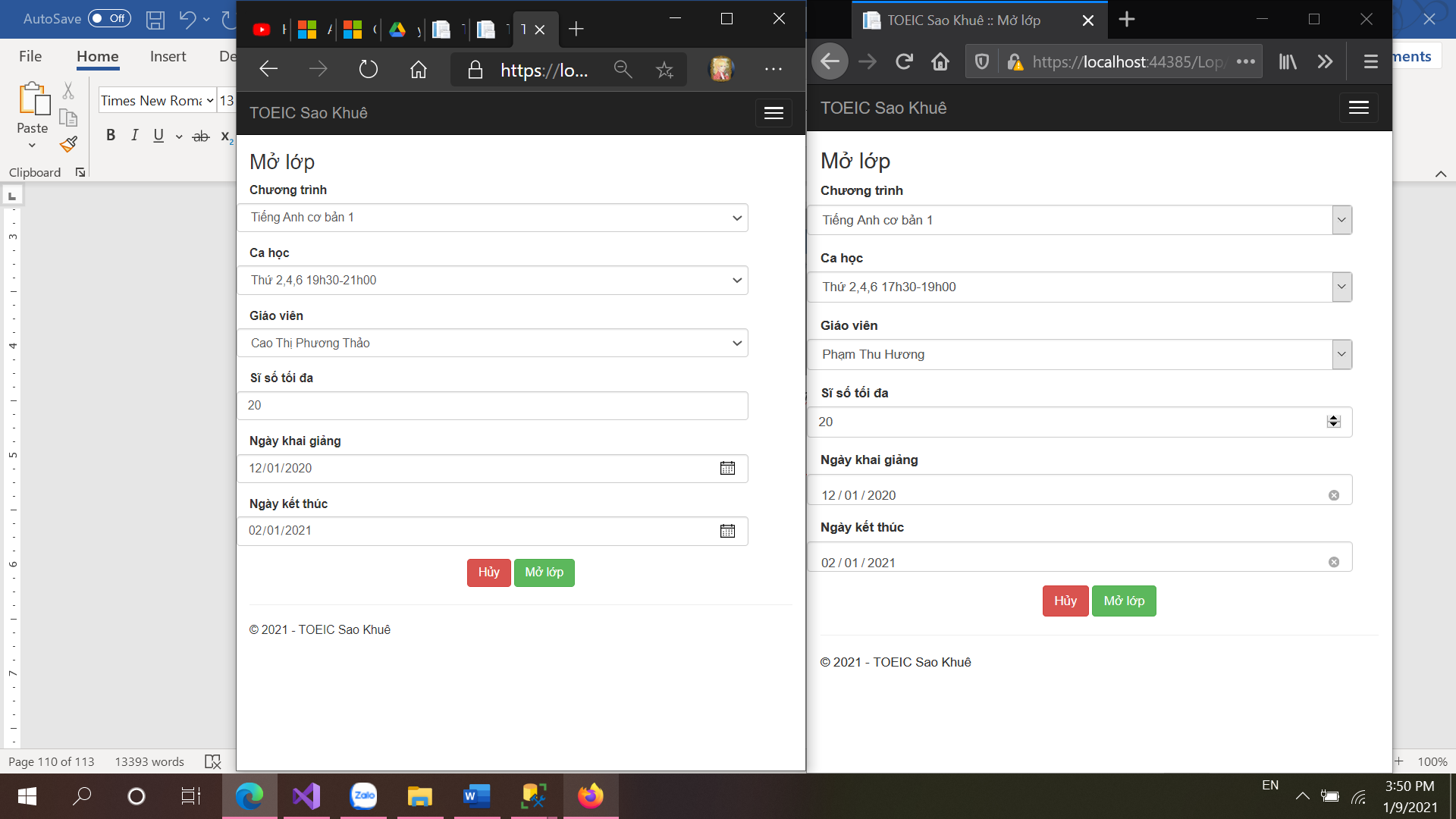
**Kết quả:** Nhân viên tư vấn lưu thông tin đăng ký trước sẽ lưu đăng ký thành công còn nhân viên tư vấn lưu thông tin đăng ký sau sẽ nhận thông báo lỗi.



Hình 27 Giả lập xử lý đồng thời - Đăng ký (3)

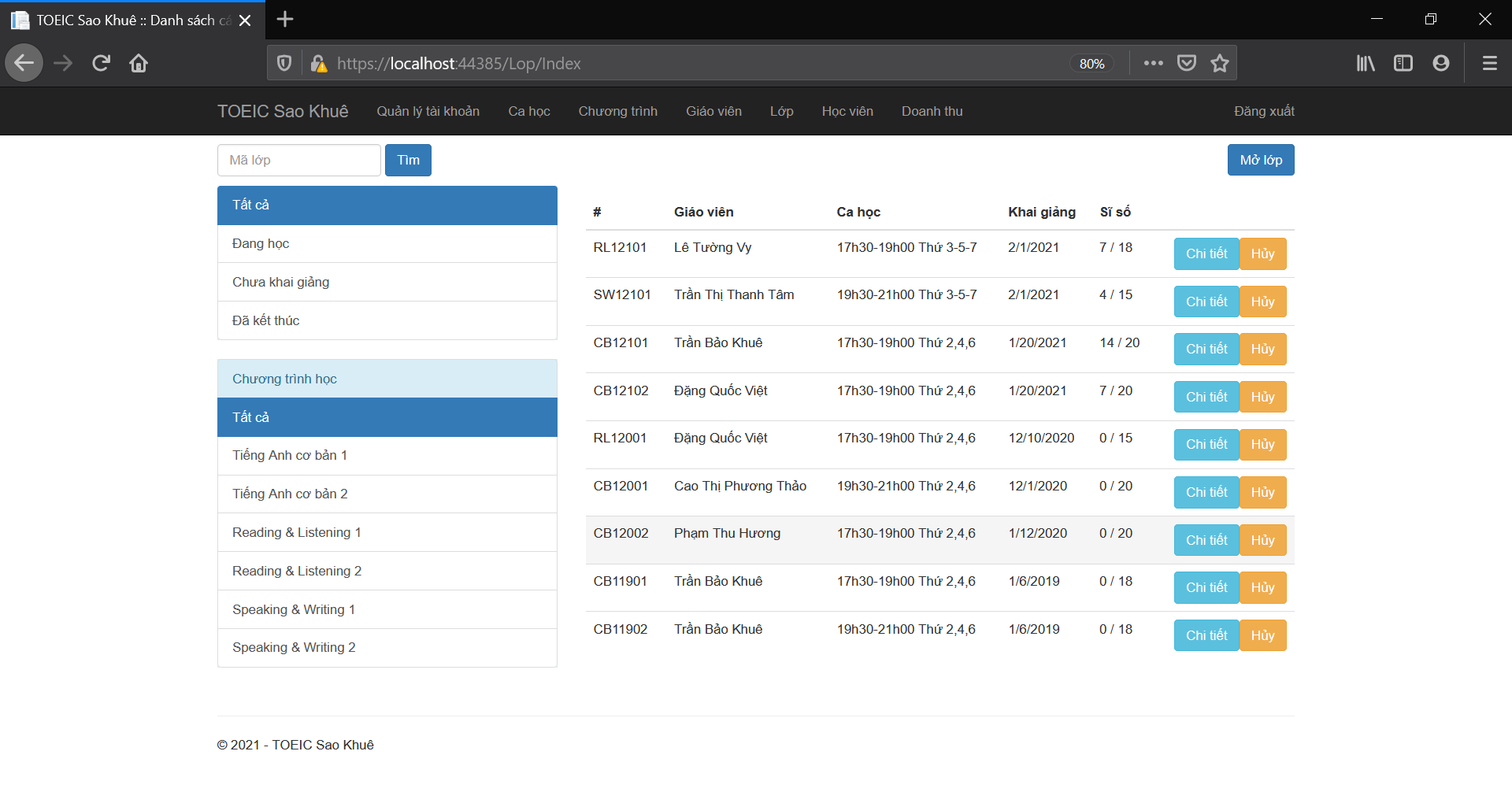
#### Mở lớp

Giả sử hai nhân viên quản lý đồng thời mở lớp cho cùng một chương trình học, cùng năm khai giảng.



Hình 28 Giả lập xử lý đồng thời - Mở lớp (1)

**Kết quả:** Cả hai lớp được mở thành công với mã lớp đúng.



Hình 29 Giả lập xử lý đồng thời - Mở lớp (2)

# KẾT LUẬN

## Kết quả đạt được

### Phía server

* Xây dựng cấu trúc dữ liệu phù hợp với yêu cầu của người dùng.
* Xây dựng các ràng buộc toàn vẹn nhằm giảm thiểu sai sót nhập liệu của người dùng.
* Xây dựng các stored procedure và function đáp ứng yêu cầu truy vấn và giao tác của người dùng, đảm bảo việc xử lý giao tác được thực hiện nhanh và có thể xử lý đồng thời.
* Phân quyền trên database.
* Lập lịch backup tự động cho database.

### Phía client

Xây dựng trang web phục vụ đầy đủ các chức năng tùy loại người dùng yêu cầu.

## Kết quả chưa đạt được

* Mã hóa dữ liệu nhằm tăng tính bảo mật.
* Giao diện chưa thực sự bắt mắt.

## Hướng phát triển trong tương lai

* Mã hóa dữ liệu để tăng tính bảo mật.
* Thiết kế giao diện bắt mắt hơn.
* Cho phép học viên đăng ký lớp online.

# TÀI LIỆU THAM KHẢO

1. Microsoft, Microsoft SQL documentation   
   https://docs.microsoft.com/vi-vn/sql/?view=sql-server-ver15
2. Microsoft, ASP.NET documentation   
   https://docs.microsoft.com/vi-vn/aspnet/core/?view=aspnetcore-5.0
3. Stack Exchange, stackoverflow.com  
   https://stackoverflow.com/

# BẢNG PHÂN CÔNG

|  |  |  |
| --- | --- | --- |
|  | **Nguyễn Sang Hoàng Kim** | **Phạm Thị Quỳnh Giang** |
| **Khảo sát nghiệp vụ** | X | X |
| **Xác định các yêu cầu cho đề tài** | X |  |
| **Thiết kế lược đồ Use case** |  | X |
| **Phân tích dữ liệu** | X |  |
| **Phân tích ràng buộc toàn vẹn** | X | X |
| **Thiết kế các bước thực hiện giao tác** | X |  |
| **Thiết kế phân quyền** | X | X |
| **Xây dựng lịch backup** |  | X |
| **Xây dựng các ràng buộc toàn vẹn và các giao tác trên server** | X | X |
| **Phân quyền trên server** | X |  |
| **Lập lịch backup tự động** |  | X |
| **Xây dựng các chức năng và giao diện cho client** | X |  |
| **Test hệ thống** | X | X |
| **Viết báo cáo** | X | X |