**Trường Đại học Đà Lạt**

**Khoa Công nghệ Thông tin**

**--- 🙡🙣 ---**



**BÁO CÁO MÔN: MẪU THIẾT KẾ**

**ĐỀ TÀI: XÂY DỰNG WEBSITE QUẢN LÝ NHÀ TRỌ**

**GVHD: Đỗ Ngọc Cường**

**Sinh viên thực hiện: 2113005 Lê Hà Hiếu Nghĩa**

**2116977 Trần Hữu Bằng**

**2115277 Nguyễn Trung Tín**

**211 Bùi Phúc Thịnh**

# MỤC LỤC

[MỤC LỤC 2](#_Toc166419136)

[LỜI CẢM ƠN 4](#_Toc166419137)

[BẢNG GIẢI THÍCH THUẬT NGỮ 4](#_Toc166419138)

[1. Lý thuyết 5](#_Toc166419139)

[1.1. Design pattern là gì? 5](#_Toc166419140)

[1.2. Lịch sử phát triển 5](#_Toc166419141)

[1.3. Các Design Pattern hiện nay 6](#_Toc166419142)

[1.3.1. Singleton 7](#_Toc166419143)

[1.3.2. Factory 7](#_Toc166419144)

[2. Hệ thống 7](#_Toc166419145)

[2.1. Tóm tắt hệ thống 7](#_Toc166419146)

[2.2. Sơ đồ ngữ cảnh của hệ thống 8](#_Toc166419147)

[2.3. Tóm tắt tính năng của hệ thống 8](#_Toc166419148)

[2.4. Sơ đồ Use Case 9](#_Toc166419149)

[2.5. Danh sách Use Cases 9](#_Toc166419150)

[2.5.1. Đăng nhập 9](#_Toc166419151)

[2.5.2. Đăng thông tin 10](#_Toc166419152)

[2.5.3. Tìm kiếm thông tin 10](#_Toc166419153)

[2.5.4. Cập nhật 10](#_Toc166419154)

[2.6. Ràng buộc và phi chức năng 12](#_Toc166419155)

[2.7. Công nghệ và công cụ sử dụng 12](#_Toc166419156)

[3. Domain Name class 12](#_Toc166419157)

[3.1. Liên kết giữa các thuộc tính 12](#_Toc166419158)

[3.2. Sơ đồ hoạt động (Activity Diagram) 13](#_Toc166419159)

[3.3. Design pattern (tạm) 13](#_Toc166419160)

[3.4. Trình tự hoạt động 14](#_Toc166419161)

[3.4.1. Đăng nhập 14](#_Toc166419162)

[3.4.2. Đăng ký 14](#_Toc166419163)

[4. Kết quả 14](#_Toc166419164)

[TÀI LIỆU THAM KHẢO 14](#_Toc166419165)

# LỜI CẢM ƠN

Lời đầu tiên chúng em xin chân thành cảm ơn giảng viên hướng dẫn, ThS.Võ Phương Bình – người hướng dẫn chúng em tận tình trong thời gian học tập và làm bài. Những sự chỉ dẫn là hướng đi chính xác để đề tài của nhóm em triển khai và hoàn thiện đáp ứng những tiêu chí đã được đưa ra.

Chúng em cũng xin chân thành cảm ơn đến quý thầy/cô Khoa Công nghệ Thông tin – Trường Đại Học Đà Lạt đã dành rất nhiều tâm huyết tận tình giảng dạy, truyền đạt những kiến thức cần thiết và tạo điều kiện tốt nhất cho em thực hiện đề tài này. Trong quá trình thực hiện bài tiểu luận, chúng em đã có cơ hội được tiếp cận với những công nghệ mới, đồng thời cũng giúp cho chúng em có thêm kinh nghiệm, nền tảng quý báu cho bản thân để tiếp tục phát triển trên con đường học tập của mình.

Dưới đây là kết quả của quá trình tìm hiểu và nghiên cứu mà nhóm em đã đạt được trong thời gian vừa qua. Trong quá trình thực hiện, do khả năng còn hạn chế và chưa có được nhiều kinh nghiệm thực tế nên chúng em không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và những góp ý chỉ bảo tận tình của quý thầy, cô.

Chúng em xin cảm ơn giảng viên ThS. Võ Phương Bình đã giảng dạy và chỉ bảo tận tình trong quá trình học tập, nghiên cứu và hoàn thành bài tiểu luận này.

Trong khuôn khổ của bài tiểu luận, do thời gian và kinh nghiệm thực tế còn hạn chế nên có những phần thực hiện chưa được tốt, chúng em rất mong nhận được sự góp ý của thầy và các bạn.

Chúng em xin chân thành cảm ơn !

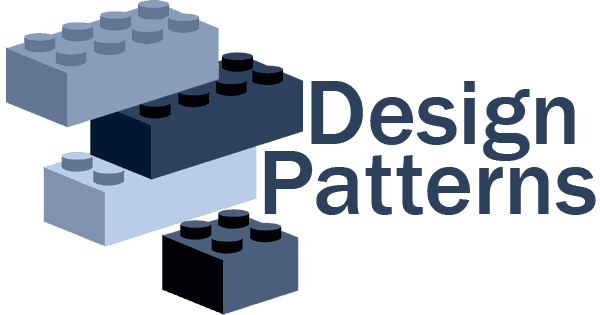
# BẢNG GIẢI THÍCH THUẬT NGỮ

|  |  |
| --- | --- |
| **Từ viết tắt** | **Tên đầy đủ** |
| **CSDL** | Cơ sở dữ liệu |
| **SinhVien** | Đối tượng sinh viên |
| **ChuTro** | Những người cho thuê trọ |
| **QuanTriVien** | Quản trị viên |
| **Tro** | Phòng trọ |
| **DSNguoiThue** | Danh sách người đến thuê trọ |

# Lý thuyết

## Design pattern là gì?

Cần nhấn mạnh rằng: **"*Design Patterns không phải là ngôn ngữ cụ thể nào cả.*"** cho nên chúng ta có thể triển khai trên nhiều ngôn ngữ lập trình khác nhau.

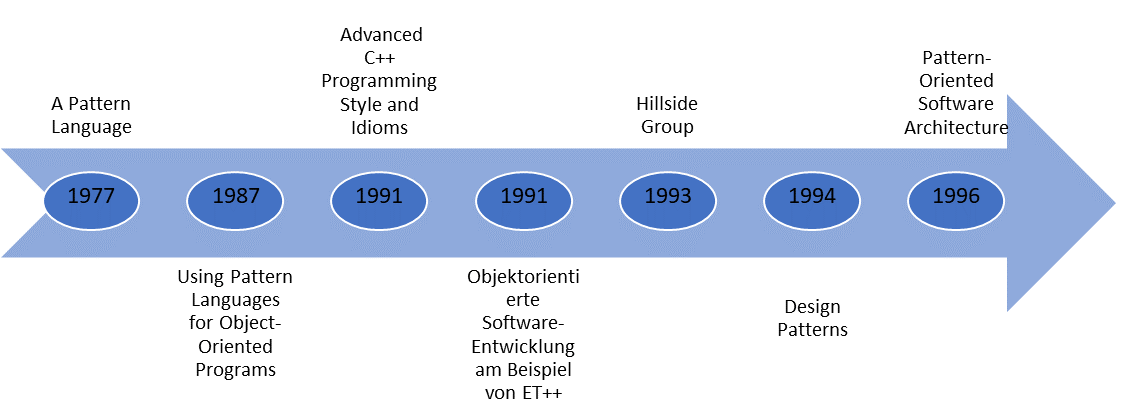


*Hình: Design pattern*

Design pattern là các giải pháp tổng thể được người dùng sử dụng trong thiết kế phần mềm. Những giải pháp này đã được tối ưu hóa, giải quyết trong các tình huống cụ thể. Nó được dùng trong lập trình hướng đối tượng, đây là kỹ thuật lập trình và có thể thực hiện ở nhiều ngôn ngữ như: Java, C#, JavaScript,… Nhờ có Design patterns giúp lập trình viên có được mẫu thiết kế giải quyết các vấn đề chung.

Lập trình viên có thể áp dụng giải pháp này để có thể giải quyết những vấn đề tương tự có  liên quan khác. Các vấn đề mà bạn thường gặp phải thì bạn sẽ có thể tự tìm ra cách để giải quyết nếu như nó chưa được xem là quá tối ưu.

## Lịch sử phát triển



*Hình: Quá trình phát triển của design pattern trong giai đoạn cuối thế kỷ 20*

Việc sử dụng "pattern" đầu tiên xuất phát từ tác phẩm của kiến ​​trúc sư (cũng là nhà nhân chủng học) Christopher Alexander. Ông viết cuốn sách "***A pattern language***" (1977), trong đó ông trình bày một ngôn ngữ mới, dựa trên những thực thể vượt thời gian được gọi là patterns. Các mẫu mô tả một vấn đề và đưa ra một giải pháp phổ quát [1] [2].

Năm 1987, Ward Cunningham và Kent Beck đã được truyền cảm hứng từ những ý tưởng của Christoph Alexander và áp dụng ý tưởng của ông vào việc phát triển phần mềm. Cùng năm đó, họ xuất bản cuốn sách "***Using Pattern Languages for Object-Oriented Programs***", một ngôn ngữ mẫu để hỗ trợ phát triển giao diện người dùng trong Smalltalk. Họ đã thiết lập hệ thống mẫu đầu tiên để thiết kế giao diện người dùng.

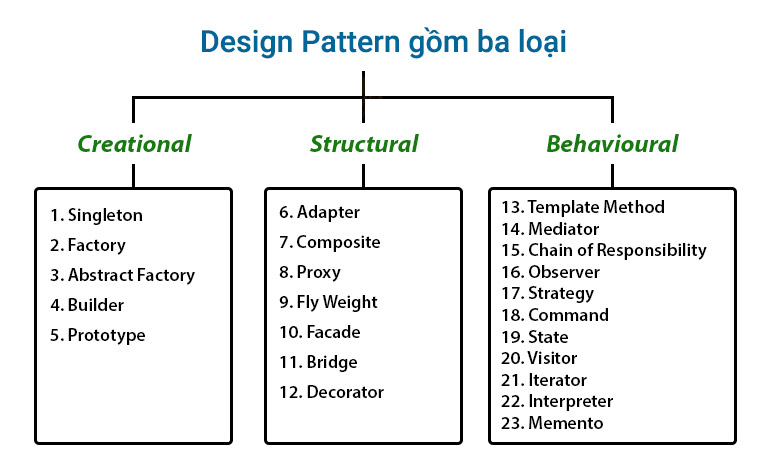
Năm 1991, James Coplien xuất bản cuốn sách " ***Advanced C++ Programming Styles and Idioms*** " [3].Cuốn sách của ông là một danh mục các thuật ngữ trong C++ như:

* Handle/Body: tách biệt giữa giao diện và cách thực hiện
* Envelope/Letter: gắn kết các ngữ nghĩa chung của giao diện và cách triển khai với nhau
* Virtual Constructor: xây dựng hệ thống phân cấp của một kiểu trừu tượng đã biết nhưng thuộc loại cụ thể chưa xác định
* Concrete Data Type: xác định đối tượng sẽ được tạo trên vùng heap của phạm vi hiện tại
* Algebraic Hierarchy: lớp cấu trúc theo các kiểu đại số
* Non-Hierarchical Addition: xử lý các phép tính số học giữa các loại không thể được thăng cấp sang loại khác

Từ năm 1995, các mẫu thiết kế phần mềm hướng đối tượng đã được chứng minh là nguồn tài nguyên quý giá cho cộng đồng công nghệ phần mềm.

## Các Design Pattern hiện nay

Hệ thống các mẫu Design pattern hiện có 23 mẫu được định nghĩa trong cuốn “***Design patterns:Elements of Reusable Object Oriented Software***” [4] và được chia thành 3 nhóm: Creational, Structural, Behavioral. Chúng được tóm tắt ở hình dưới:



*Hình: Phân loại design pattern theo mục đích sử dụng*

Chi tiết về các pattern (chỉ liệt kê những pattern được áp dụng trong dự án nhóm)

### Singleton

Khi một lớp được thiết kế theo Singleton sẽ có phương thức trả về duy nhất chính là biểu hiện của nó. Nếu dấu hiệu đó chưa tồn tại thì nó sẽ được tự động tạo mới trước khi trả về. Quá trình này thường được thực hiện bằng cách sử dụng biến tĩnh hoặc biến toàn cục để lưu trữ phương thức tĩnh, hỗ trợ truy cập đến biểu tượng đó. Nó đảm bảo rằng một class chỉ có duy nhất một instance.

Trong các trường hợp cần quản lý tài nguyên có giới hạn như cơ sở kết nối cơ sở dữ liệu, Singleton góp phần vào việc đảm bảo khả năng sử dụng được dùng để tiết kiệm tài nguyên.

### Factory

Factory Method cung cấp một interface, phương thức trong việc tạo nên một đối tượng (object) trong class. Nhưng để cho class con kế thừa của nó có thể ghi đè để chỉ rõ đối tượng (object) nào sẽ được tạo. Factory method giao việc khởi tao một đối tượng (object) cụ thế cho lớp con (subclass)

Nhiệm vụ của Factory Pattern là quản lý và trả về các đối tượng theo yêu cầu, giúp cho việc khởi tạo đổi tượng một cách linh hoạt hơn. Ngoài ra, đây đích thực là một nhà máy theo đúng như cái tên, và nhà máy này sẽ “sản xuất” các đối tượng theo yêu cầu. [5].

1. Hiểu rõ các yêu cầu
2. Viết các test case
   * Quản lý kiểm tra: HP-ALM hoặc testlink: có thể tự viết ra các kịch bản kiểm thử, các test case và test scripting, có thêt liên kết yêu cầu của dự án. Nó còn đảm bảo các báo cáo thử nghiệm được dễ dàng và hiệu quả.
   * Quản lý lỗi: Bugzilla, Mantis: đây là cần thiết để nâng cao và quản lý các lỗi. Đôi khi bạn phải tạo các báo cáo lỗi như đã nêu ở bản phát hành(từ trước)

Ưu và nhược điểm của kiểm thử Hộp đen theo tác giả Isha Sunita Sangwan [6]:

# Hệ thống

## Tóm tắt hệ thống

Hệ thống nhà trọ Đà Lạt là một trang web tìm nhà trọ, các căn hộ hoặc nhà nguyên căn nhằm đáp ứng cho các sinh viên cũng như những người lao động từ những tỉnh thành khác trên cả nước khi đến để làm việc và học tập. Trang web giúp người dùng tiết kiệm được thời gian cũng như kinh phí sau khi sử dụng trang web.

## Sơ đồ ngữ cảnh của hệ thống

A diagram of a diagram

Description automatically generated

***Hình 1.1 – Mô tả ngữ cảnh của hệ thống***

## Tóm tắt tính năng của hệ thống

**Bảng: Tóm tắt tính năng của trang web**

|  |  |
| --- | --- |
| **ID** | **Mô tả tính năng** |
| **FEA-01** | **Đăng nhập** |
| **FEA-02** | **Đăng ký** |
| **FEA-03** | **Đăng thông tin liên quan** |
| **FEA-04** | **Tìm kiếm thông tin** |
| **FEA-05** | **Cập nhật thông tin của SinhVien, do ChuTro thực hiện** |

## Sơ đồ Use Case

**A diagram of a network

Description automatically generated**

***Hình 1.2 – Sơ đồ Use Case***

Mối quan hệ giữa các use case

## Danh sách Use Cases

### Đăng nhập

|  |  |
| --- | --- |
| **Use Case UC1: Đăng nhập** | |
| **Phạm vi truy cập** | Tất cả các actor |
| **Luồng sự kiện chính** | 1. Hệ thống hiển thị màn hình đăng nhập 2. Người dùng nhập username, password 3. Nếu thành công, hệ thống thể hiện việc đăng nhập đã hoàn tất 4. Kết thúc use case |
| **Luồng sự kiện phụ** | **1a** Mật khẩu không hợp lệ (khi nhập sai username hoặc password   * Bắt nhập lại thông tin, đi kèm đó là thông báo nhập sai và yêu cầu nhập lại * Quay lại bước 2 trong luồng sự kiện chính   **1b.** Quên mật khẩu: khi người dùng chọn chức năng quên mật khẩu trên màn đăng nhập   * Hiển thị màn hình |

### Đăng thông tin

|  |  |
| --- | --- |
| **Use Case UC2: Đăng thông tin** | |
| **Phạm vi truy cập** | ChuTro |
| **Luồng sự kiện chính** | 1a: Tạo mới   1. Thực hiện việc đăng nhập 2. Đưa ra form chứa các textbox liên quan, người chủ chỉ cần nhập thông tin liên quan vào 3. Nhấn xác nhận 4. Gửi thông báo thành công, thông tin mới đó được hiển thị lên form 5. Kết thúc thao tác (có thể load lại trang) |
| **Luồng sự kiện phụ** | * Nếu chưa đăng nhập thì phải thực hiện lại bước **1** trong luồng sự kiện chính |

### Tìm kiếm thông tin

|  |  |
| --- | --- |
| **Use Case UC3: Tìm kiếm thông tin** | |
| **Phạm vi truy cập** | SinhVien |
| **Luồng sự kiện chính** | 1a:   1. Truy cập trang web |
| **Luồng sự kiện phụ** | * Có thể không cần đăng nhập |

### Cập nhật

|  |  |
| --- | --- |
| **Use Case UC4: Cập nhật( Thêm xóa, sửa)** | |
| **Phạm vi truy cập** | ChuTro |
| **Điều kiện kiên quyết** | * Phải hoàn thành việc đăng nhập vào web |
| **Luồng sự kiện chính** | 1a: Chức năng thêm   1. Hệ thống hiển thị màn hình quản lý (điền thông tin của người mới đến thuê) 2. ChuTro nhập thông tin liên quan 3. Nếu thành công, hiển thị thêm thông tin thành công, cập nhật trong db 4. Kết thúc thao tác   1b: Chức năng chỉnh sửa hoặc xóa   1. Hệ thống hiển thị màn hình quản lý 2. Nhấn vào nút edit hoặc delete ở góc bên phải 3. Hiển thị thông tin để user có thể thực hiện thao tác 4. Hệ thống gửi form xác nhận có muốn thay đổi hay không? 5. Nếu thành công, hiển thị thông tin đã chỉnh sửa hoặc xóa thành công 6. Kết thúc thao tác |
| **Luồng sự kiện phụ** |  |

## Ràng buộc và phi chức năng

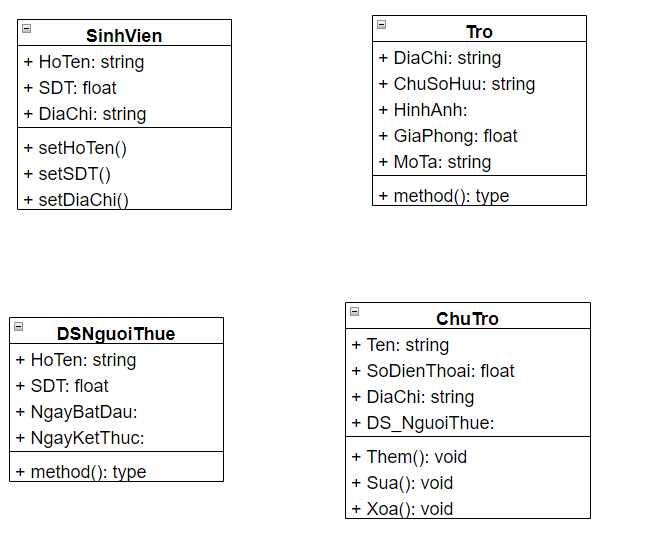
## Công nghệ và công cụ sử dụng

* Dự án được viết bằng ngôn ngữ ReactJS, sử dụng CSDL MongoDB
* Môi trường phát triển: Visual Studio Code

# Domain Name class

## Liên kết giữa các thuộc tính

|  |
| --- |
| **Mô tả các quan hệ** |
| Một **SinhVien** có thể đến thuê tại một phòng trong một khoảng thời gian nhất định. |
| Một **chủ trọ** có thể cho nhiều **SinhVien** thuê tại một khu trọ |
| Một **chủ trọ** có thể sở hữu nhiều khu trọ (nếu có thể) |
| Một **ChuTro** chỉ có thể có được duy nhất một **DSNguoiThue** |
| Một **Tro** có thể được nhiều người sử dụng trong nhiều lần, và chỉ được sở hữu bởi duy nhất một **ChuTro** |



***Hình: Mô hình miền của các lớp với liên kết và thuộc tính***

## Sơ đồ hoạt động (Activity Diagram)

## Design pattern (tạm)

<https://200lab.io/blog/giai-thich-cac-ky-hieu-trong-class-diagram/#2-association>

<https://viblo.asia/p/tong-hop-cac-bai-huong-dan-ve-design-pattern-23-mau-co-ban-cua-gof-3P0lPQPG5ox>

<https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method/>

* Tạo enum chứa các loại boarding house

Chúng ta có một super class với nhiều class con và dựa trên đầu vào, chúng ta cần trả về một class con. Mô hình này giúp chúng ta đưa trách nhiệm của việc khởi tạo một lớp từ phía người dùng (client) sang lớp Factory. Chúng ta không biết sau này sẽ cần đến những lớp con nào nữa. Khi cần mở rộng, hãy tạo ra sub class và implement thêm vào factory method cho việc khởi tạo sub class này.

<https://gpcoder.com/4190-huong-dan-java-design-pattern-singleton/>

## Trình tự hoạt động

### Đăng nhập



***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

### Đăng ký

***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

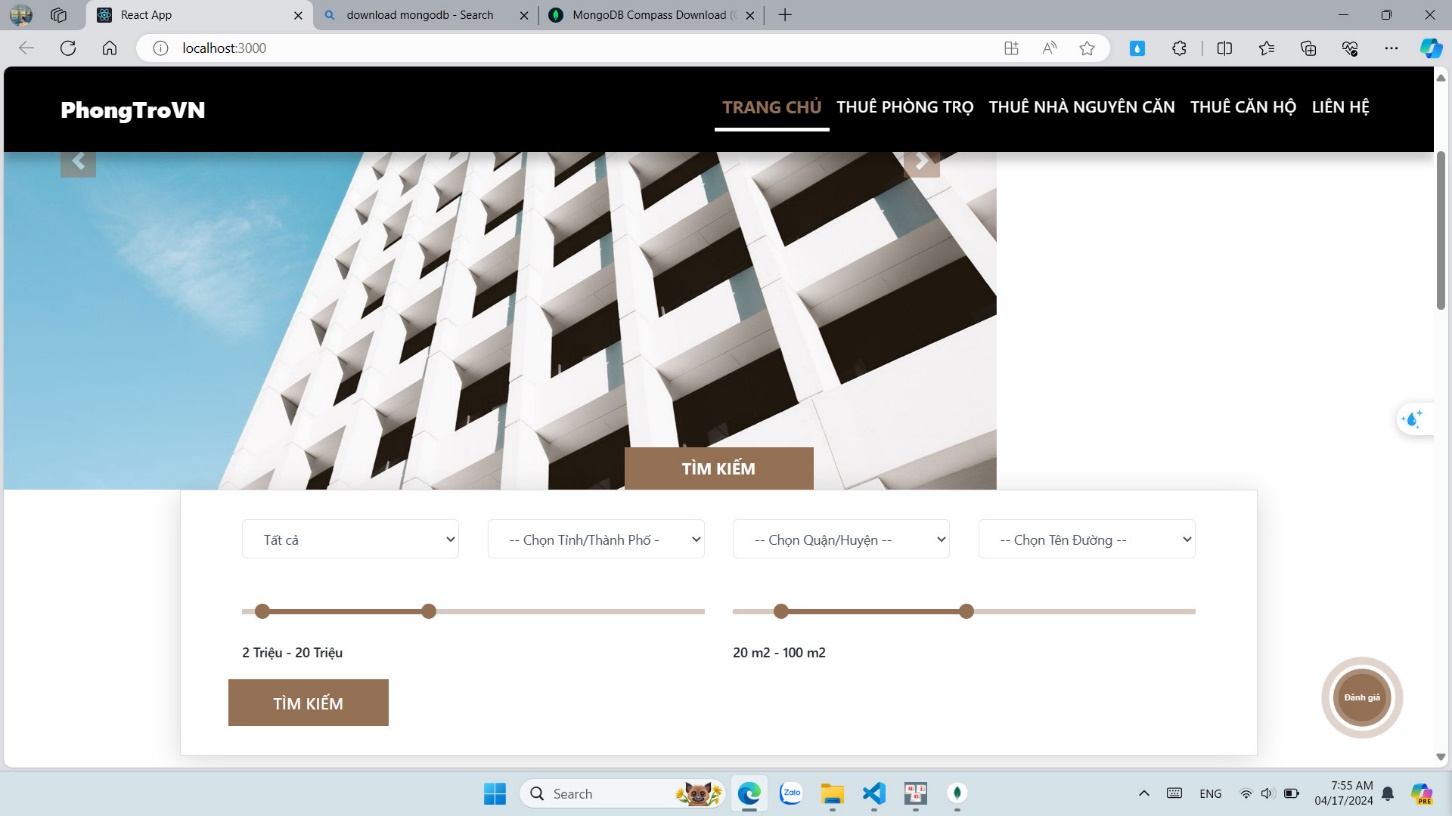
***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

***Hình: Trình tự hoạt động đối với thao tác đăng nhập***

# Kết quả

Giao diện chính



# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Sablier, “Origin of design patterns - Coding Game,” Coding Game, 21 7 2023. [Trực tuyến]. Available: https://www.codingame.com/playgrounds/503/design-patterns/origin-of-design-patterns#:~:text=The%20first%20use%20of%20%22patterns,and%20offer%20an%20universal%20solution.. |
| [2] | S. I. M. S. M. J. I. F.-K. S. A. Christopher Alexander, A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series), Oxford: Oxford University Press, 1977. |
| [3] | J. O. Coplien, Advanced C++ Programming Styles and Idioms, Addison-Wesley Professional, 1991. |
| [4] | Erich Gamma, Richard Helm, Ralph Johnson và John Vlissides, Design patterns: Elements of Reusable Object Oriented Software, 1st biên tập viên, Zürich: Wiley, 2023. |
| [5] | G. Coder, “Hướng dẫn Java Design Pattern – Factory Method,” 12 9 2018. [Trực tuyến]. Available: https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method/. |
| [6] | S. S. Isha, “Software Testing Techniques and Strategies,” tập 4, số 4, pp. 99-102, 4 2014. |