**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**MÔN HỌC: ĐỒ ÁN PHÁT TRIỂN ỨNG DỤNG WEB**

**ĐỀ TÀI**

**QUẢN LÝ KÝ TÚC XÁ K7**

**GVBM: Bùi Chí Thành**

**Thành viên nhóm:**

|  |  |  |
| --- | --- | --- |
| **1.** | **Trần Quốc Trung** | **62132908** |
| **2.** | **Tô Văn Phơ** | **62131549** |
| **3.** | **Trần Anh Tuấn** | **62132540** |

# MỤC LỤC

[MỤC LỤC 2](#_Toc136811584)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 4](#_Toc136811585)

[1.1. Lý do chọn đề tài: 4](#_Toc136811586)

[1.2. Mục tiêu của đề tài 4](#_Toc136811587)

[1.3. Giới hạn và phạm vi của đề tài 4](#_Toc136811588)

[1.4. Tính khả thi 5](#_Toc136811589)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc136811590)

[2.1. Tổng quan về ngôn ngữ Asp.NET MVC 6](#_Toc136811591)

[2.1.1. Giới thiệu Asp.NET MVC Framework 6](#_Toc136811592)

[2.1.2 Lý do chọn Asp.NET MVC 11](#_Toc136811593)

[2.2. Tổng quan về SQL SERVER 12](#_Toc136811594)

[2.2.1. Hệ quản trị CSDL SQL Server 12](#_Toc136811595)

[**2.2.2. Đặc điểm của SQL Server và đối tượng làm việc** 13](#_Toc136811596)

[CHƯƠNG 3: NỘI DUNG THỰC HIỆN 14](#_Toc136811597)

[3.1 . Đối tượng sử dụng website: 14](#_Toc136811598)

[3.2. Các chức năng chính của website: 14](#_Toc136811599)

[3.3 Phân tích thiết kế hệ thống 15](#_Toc136811600)

[3.3.1 Các yêu cầu chức năng chính của hệ thống 15](#_Toc136811601)

[**3.3.2** **Các biểu đồ Use Case** 16](#_Toc136811602)

[3.4 Thiết kế cơ sở dữ liệu 18](#_Toc136811603)

[3.4.1 Mô hình cơ sở dữ liệu 18](#_Toc136811604)

[3.6.2 Danh sách các lớp đối tượng 18](#_Toc136811605)

[3.5 Thiết kế và đặc tả giao diện 21](#_Toc136811606)

[3.5.1 Giao diện trang đăng nhập 21](#_Toc136811607)

[3.5.2 Giao diện trang đăng ký 22](#_Toc136811608)

[3.5.3 Giao diện trang danh sách phòng ở 23](#_Toc136811609)

[3.5.4 Giao diện trang danh sách sinh viên 24](#_Toc136811610)

[*Hình 8: Giao diện trang chi trang danh sách sinh viên* 24](#_Toc136811611)

[3.5.5 Giao diện dịch vụ 24](#_Toc136811612)

[3.5.6 Giao diện trang quản lý sinh viên 25](#_Toc136811613)

[3.5.7 Giao diện trang thêm, sửa, xóa sinh viên 26](#_Toc136811614)

[3.5.8 Giao diện trang quản lý phòng ở 28](#_Toc136811615)

[3.5.9 Giao diện trang quản lý dịch vụ 28](#_Toc136811616)

[3.5.10 Giao diện trang quản lý dịch vụ phòng 29](#_Toc136811617)

[3.5.10 Giao diện trang quản lý hóa đơn dịch vụ 30](#_Toc136811618)

[CHƯƠNG 4: KẾT LUẬN 31](#_Toc136811619)

[4.1 Kết quả đạt được của đề tài 31](#_Toc136811620)

[4.2 Hạn chế của đề tài 31](#_Toc136811621)

[4.3 Kết luận 31](#_Toc136811622)

[**TÀI LIỆU THAM KHẢO** 32](#_Toc136811623)

# TỔNG QUAN VỀ ĐỀ TÀI

## Lý do chọn đề tài:

Đây là một đề tài rất thú vị và hữu ích, bởi vì quản lý kí túc xá là một công việc phức tạp, đòi hỏi nhiều kỹ năng và trách nhiệm. Một hệ thống quản lý kí túc xá hiệu quả giúp đảm bảo việc cung cấp dịch vụ lưu trú an toàn, tiện nghi và tốt nhất cho các sinh viên. Ngoài ra, hệ thống còn giúp quản lý thu chi, bảo trì và sửa chữa, quản lý tiện ích và dịch vụ, giúp tối ưu hóa hoạt động của kí túc xá. Với số lượng sinh viên ngày càng tăng, đề tài này còn càng trở nên cấp thiết và có triển vọng phát triển.

## Mục tiêu của đề tài

* Trau dồi kiến thức và kỹ năng về quản lý, thiết kế và triển khai hệ thống quản lý kí túc xá.
* Tăng cường khả năng làm việc nhóm và phát triển tư duy tổng quan về vấn đề.
* Tối ưu hoá quy trình quản lý tại kí túc xá, giảm thiểu sự cố và tăng tính hiệu quả.
* Xây dựng một website quản lý kí túc xá với giao diện thân thiện, dễ sử dụng, đáp ứng các nhu cầu của người dùng. Các tính năng chính bao gồm: quản lý phòng, quản lý sinh viên, thanh toán, bảo trì và sửa chữa, giúp nâng cao trải nghiệm lưu trú của sinh viên.

## Giới hạn và phạm vi của đề tài

* Đối tượng triển khai: Đề tài sẽ triển khai tại một số kí túc xá nhỏ, tầm khoảng 30-100 phòng ở các trường đại học hoặc cao đẳng tại Việt Nam.
* Đối tượng sử dụng: Phần mềm sẽ được sử dụng bởi các nhân viên quản lý kí túc xá, giúp họ quản lý thông tin sinh viên, phòng trọ, quản lý thu chi, bảo trì và sửa chữa, cũng như cung cấp các dịch vụ tiện ích cho sinh viên.
* Phạm vi tính năng: Phần mềm sẽ tập trung vào các tính năng quản lý phòng, quản lý sinh viên, thanh toán, bảo trì và sửa chữa, giúp quản lý kí túc xá hiệu quả hơn. Ngoài ra, phần mềm sẽ hỗ trợ việc tạo và quản lý các thông báo, cập nhật thông tin, và các chức năng báo cáo tổng hợp để hỗ trợ quản lý kí túc xá đưa ra quyết định.

## Tính khả thi

* Xây dựng website quản lý kí túc xá: Với sự phát triển của công nghệ web hiện nay, việc xây dựng website quản lý kí túc xá là khả thi. Có nhiều công cụ, ngôn ngữ lập trình và framework hỗ trợ việc xây dựng website, giúp cho việc phát triển và triển khai phần mềm được nhanh chóng và dễ dàng hơn.
* Xây dựng hệ thống quản lý kí túc xá: Tính khả thi của việc xây dựng hệ thống quản lý kí túc xá phụ thuộc vào độ phức tạp của tính năng và yêu cầu của phần mềm. Tuy nhiên, với việc sử dụng các công nghệ hiện đại và phù hợp, việc xây dựng hệ thống quản lý kí túc xá là khả thi.
* Các công nghệ sử dụng: Việc lựa chọn các công nghệ sử dụng phù hợp sẽ giúp cho đề tài trở nên khả thi hơn. Ví dụ, sử dụng ngôn ngữ lập trình phù hợp, cơ sở dữ liệu đáp ứng được yêu cầu, các framework hỗ trợ phát triển phần mềm nhanh và hiệu quả.
* Tài nguyên: Việc có đủ tài nguyên, bao gồm tài chính, nhân lực và thiết bị để triển khai đề tài là cần thiết. Tuy nhiên, với quy mô nhỏ và phù hợp, việc triển khai đề tài là khả thi.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về ngôn ngữ Asp.NET MVC

Mẫu kiến trúc Model - View - Controller được sử dụng nhằm chi ứng dụng thành ba thành phần chính: model, view và controller. Nền tảng ASP.NET MVC giúp cho chúng ta có thể tạo được các ứng dụng web áp dụng mô hình MVC thay vì tạo ứng dụng theo mẫu ASP.NET Web Forsm. Nền tảng ASP.NET MVC có đặc điểm nổi bật là nhẹ (lighweigt), dễ kiểm thử phần giao diện (so với ứng dụng Web Forms), tích hợp các tính năng có sẵn của ASP.NET. Nền tảng ASP.NET MVC được định nghĩa trong namespace System.Web.Mvc và là một phần của name space System.Web.

MVC là một mẫu thiết kế (design pattern) chuẩn mà nhiều lập trình viên đã quen thuộc. Một số loại ứng dụng web sẽ thích hợp với kiến trúc MVC. Một số khác vẫn thích hợp với ASP.NET Web Forms và cơ chế postbacks. Đôi khi có những ứng dụng kết hợp cả hai kiến trúc trên.

2.1.1. Giới thiệu Asp.NET MVC Framework

ASP.NET là một nền tảng ứng dụng web (web application framework) được phát triển và cung cấp bởi Microsoft, cho phép những người lập trình tạo ra những trang web động, những ứng dụng web và những dịch vụ web. - Dựa trên ASP.NET, ASP.NET MVC cho phép các nhà phát triển phần mềm xây dựng một ứng dụng web dựa trên mẫu thiết kế MVC. - MVC là một mẫu thiết kế (design pattern) chuẩn, được sử dụng nhằm chia ứng dụng thành ba thành phần chính: model, view và controller. - Phiên bản ASP.NET MVC: ASP.NET MVC 5 (10/2013), ASP.NET MVC 5.2.7 (11/2017).

Nền tảng MVC bao gồm các thành phần dưới đây :

**Hình 1: Mẫu Model - View - Controller**

**Models:** Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong CSDL. Ví dụ như, một đối tượng Product (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Products ở SQL Server.

Trong các ứng dụng nhỏ, model thường là chỉ là một khái niệm nhằm phân biệt hơn là được cài đặt thực thụ, ví dụ, nếu ứng dụng chỉ đọc dữ liệu từ CSDL và gởi chúng đến view, ứng dụng không cần phải có tầng model và các lớp liên quan. Trong trường hợp này, dữ liệu được lấy như là một đối tượng model (hơn là tầng model).

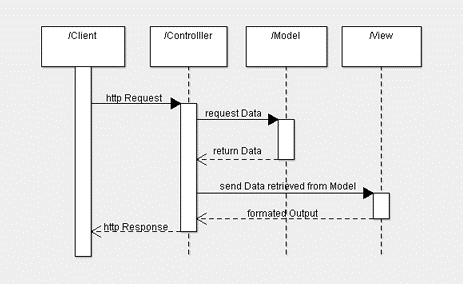
**Views:**  là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Ví dụ như, view dùng để cập nhật bảng Products sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Product.

**Controllers**: Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng. Ví dụ, controller sẽ quản lý các dữ liệu người dùng gởi lên (query-string values) và gởi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này.

Mẫu MVC giúp bạn tạo được các ứng dụng mà chúng phân tách rạch ròi các khía cạnh của ứng dụng (logic về nhập liệu, logic xử lý tác vụ và logic về giao diện). Mẫu MVC chỉ ra mỗi loại logic kể trên nên được thiếp lập ở đâu trên ứng dụng. Logic giao diện (UI logic) thuộc về views. Logic nhập liệu (input logic) thuộc về controller. Và logic tác vụ (Business logic – là logic xử lý thông tin, mục đích chính của ứng dụng) thuộc về model. Sự phân chia này giúp bạn giảm bớt được sự phức tạp của ứng dụng và chỉ tập trung vào mỗi khía cạnh cần được cài đặt ở mỗi thời điểm. Ví dụ như bạn chỉ cần tập trung vào giao diện (views) mà không phải quan tâm đến logic xử lý thông tin của ứng dụng.

Để quản lý sự phức tạp của ứng dụng, mẫu MVC giúp cho chúng ta có thể kiểm thử ứng dụng dễ dàng hơn hẳn so với khi áp dụng mẫu Web Forms. Ví dụ, trong một ứng dụng ASP.NET Web Forms, một lớp thường được sử dụng để hiển thị thông tin xuất ra cho người dùng và đồng thời xử lý thông tin người dùng nhập. Việc xây dựng các bộ test tự động cho ứng dụng Web Forms là rất phức tạp, bởi để kiểm thử mỗi trang web, bạn phải khởi tạo đối tượng trang, khởi tạo tất cả các control được sử dụng trong trang và các lớp phụ thuộc trong ứng dụng. Và bởi vì có quá nhiều lớp cần được khởi tạo để chạy được trang, thật khó để có thể viết các test chỉ tập trung vào một khía cạnh nào đó của ứng dụng. Và vì thế, kiểm thử đối với các ứng dụng dứa trên nền tảng Web Forms sẽ khó khăn hơn nhiều so với khi áp dụng trên ứng dụng MVC. Hơn thế nữa, việc kiểm thử trên nền tảng Web Forms yêu cầu phải sử dụng đến web server. Nền tảng MVC phân tách các thành phần và sử dụng các interface (khái niệm giao diện trong lập trình hướng đối tượng), và nhờ đó có thể kiểm thử các thành phần riêng biệt trong tình trạng phân lập với các yếu tố còn lại của ứng dụng

Sự phân tách rạch ròi ba thành phần của ứng dụng MVC còn giúp cho việc lập trình diễn ra song song. Ví dụ như một lập trình viên làm việc với view, lập trình viên thứ hai lo cài đặt logic của controller và lập trình viên thứ ba có thể tập trung vào logic tác vụ của model tại cùng một thời điểm.



Hình 2: Mô hình tuần tự của MVC

Lấy ví dụ một GUI Component (thành phần đồ họa người dùng ) đơn giản là checkbox. Checkbox có thành phần Model để quản lý trạng thái của nó là check hay uncheck, thành phần View để thể hiện nó với trạng thái tương ưng lên màn hình, và thành phần Controller để xử lý nhưng sự kiện khi có sự tương tác của người sử dụng hoặc các đối tượng khác lên checkbox.

Khi ngươi sử dụng nhần chuột vào Check box , thành phần Controller của Checkbox sẽ xử lý sự kiện này, yêu cầu thành phần Model thay đổi dữ liệu trạng thái. Sau khi thay đổi trạng thái, thành phần Model phát thông điệp đến thành phần View và Controller. Thành phần View của Checkbox nhận được thông điệp sẽ cập nhật lại thể hiện của Checkbox, phản ánh chính sác trạng thái Checkbox do Model lưu giữ. Thành phần Controller nhận được thông điệp do Model gởi tới sẽ có nhưng tương tác phản hòi với người sử dụng nếu cần thiết.

**Tiềm hiểu thêm về Controler**

Controller có trách nhiệm chính là điều hướng các yêu cầu của người sử dụng. Như vậy trên toàn ứng dụng của ta, tất cả các request đều sẽ phải đi tới controller. Và tại đây, ứng với các tham số người sử dụng truyền mà ta đưa họ đến một tác vụ nào đó trên ứng dụng.

Tại các tác vụ này, chúng sẽ thông qua lớp model để làm việc và trả kết quả trở về controller. Cuối cùng controller sẽ đẩy dữ liệu thao tác tới view. View là thành phần cuối cùng mà người sử dụng nhận được khi họ giở request tới ứng dụng.

Có thể hiểu controller, giống với kỹ thuật đa cấp với các tác vụ chạy ứng dụng phân cấp theo từng nhánh riêng biệt như: Module, action,… Qua hình vẽ này, ta hiểu rằng. Để có thể thao tác với các action (hành động) ta cần phải đi qua file index.php. Lúc này file index đóng vai trò như một controller được dùng để điều hướng các request. Ứng với các request thì nó sẽ trả về một controller khác để xử lý tác vụ một cách cụ thể. Tại controller con, nó sẽ gọi các action riêng biệt. Ở đó, thông qua các action mà nó gọi tới các file xử lý giản đơn.

**Tìm hiểu Model**

Model là thành phần chủ yếu được sử dụng để thao tác xử lý dữ liệu. Trong các framework, Model vẫn thường sử dụng theo phương thức Active Record. Một trong những design pattern. Chúng có tác dụng rút ngắn thời gian viết câu truy vấn cho người sử dụng. Biến những câu truy vấn phức tạp trở nên gần gũi và thân thiện với người sử dụng thông qua các thư viện được định nghĩa sẵn.

Model thường sẽ là các phương thức có trách nhiệm xử lý các tác vụ như: select, insert, update, delete các record trong database. Ứng với các lấy dữ liệu, model thường sử dụng mảng để gởi trả kết quả về. Vì mảng có thể cho phép model lưu trữ nhiều thông tin hơn, nên thường các record khi bóc tách chúng sẽ mang các dữ liệu của database một cách chi tiết.  
Khi sử dụng models, ta cũng cần tuân theo nguyên tắc chính của chúng là không xuất giá trị trực tiếp trong model. Mà tất cả những dữ liệu ấy, phải đưa vào mảng và trả về theo phương thức. Và tiếp tục ở view ta sẽ sử dụng nó để lấy dữ liệu ra.

**Tìm hiều về View**

View là phần hiển thị thông tin tương phản khi gởi và nhận request. Trước đây, khi người lập trình chưa nghĩ tới view. Họ thường thao tác xử lý dữ liệu ngay trực tiếp trên ứng dụng và đổ cả dữ liệu ngay trên file PHP đó. Điều này làm cho ứng dụng trở nên cồng kềnh, và đặc biệt rất khó cho việc bảo trì nâng cấp sau này. Nhất là đối với designer, việc thay đổi giao diện của một website luôn làm cho họ cảm thấy đau đầu vì phải vọc thẳng vào core. Trước đây, để giải quyết tình huống này. Người ta thường sử dụng template để phân tách website thành 2 mảng riêng biệt. Một là giao diện và một là core. Việc chỉnh sửa giao diện trở nên đơn giản hơn đối với họ so với cách viết thập cẩm kia. Tuy nhiên, các thư viện này thực chất sẽ làm cho ứng dụng của chúng ta trở nên chậm chạp hơn bao giờ hết. Bởi chúng phải phiên dịch nhiều lần các kịch bản.

Chẳng hạn: Để dễ thao tác, smarty sẽ dịch ngược các yêu cầu của bạn sang ngôn ngữ của nó. Sau đó chúng sẽ chuyển ngôn ngữ đó sang PHP và thao tác xử lý trên nó.  
Việc này sẽ làm ứng dụng chậm chạp, do cứ phải dịch qua, dịch lại một kịch bản. Trong khi, với sự kết hợp của PHP thuần, ứng dụng của bạn sẽ nhanh và ổn định hơn nhiều. Và view cũng là một phần trong việc nâng cấp những hạn chế ấy. Chúng giúp giảm thiểu tối đa quá trình biên dịch nhiều lần. Và làm cho ứng dụng trở nên mạnh mẽ và chuyên nghiệp hơn nhiều so với cách lập trình thuần.

2.1.2 Lý do chọn Asp.NET MVC

* Các tính năng của Asp.NET MVC:
* Tách bạch các tác vụ của ứng dụng (logic nhập liệu, business logic, và logic giao diện), dễ dàng kiểm thử và mặc định áp dụng hướng phát triển TDD. Tất cả các tính năng chính của mô hình MVC được cài đặt dựa trên interface và được kiểm thử bằng cách sử dụng các đối tượng mocks, mock object là các đối tượng mô phỏng các tính năng của những đối tượng thực sự trong ứng dụng. Bạn có thể kiểm thử unit-test cho ứng dụng mà không cần chạy controller trong tiến trình ASP.NET, và điều đó giúp unit test được áp dụng nhanh chóng và tiện dụng. Bạn có thể sử dụng bất kỳ nền tảng unit-testing nào tương thích với nền tảng .NET.
* MVC là một nền tảng khả mở rộng (extensible) & khả nhúng (pluggable). Các thành phần của ASP.NET MVC được thiết kể để chúng có thể được thay thế một cách dễ dàng hoặc dễ dàng tùy chỉnh. Bạn có thể nhúng thêm view engine, cơ chế định tuyến cho URL, cách kết xuất tham số của action-method và các thành phần khác. ASP.NET MVC cũng hỗ trợ việc sử dụng Dependency Injection (DI) và Inversion of Control (IoC). DI cho phép bạn gắn các đối tượng vào một lớp cho lớp đó sử dụng thay vì buộc lớp đó phải tự mình khởi tạo các đối tượng. IoC quy định rằng, nếu một đối tượng yêu cầu một đối tượng khác, đối tượng đầu sẽ lấy đối tượng thứ hai từ một nguồn bên ngoài, ví dụ như từ tập tin cấu hình. Và nhờ vậy, việc sử dụng DI và IoC sẽ giúp kiểm thử dễ dàng hơn.
* ASP.NET MVC có thành phần ánh xạ URL mạnh mẽ cho phép bạn xây dựng những ứng dụng có các địa chỉ URL xúc tích và dễ tìm kiếm. Các địa chỉ URL không cần phải có phần mở rộng của tên tập tin và được thiết kế để hỗ trợ các mẫu định dạng tên phù hợp với việc tối ưu hóa tìm kiếm (URL) và phù hợp với lập địa chỉ theo kiểu REST.
* Hỗ trợ sử dụng đặc tả (các thẻ) của các trang ASP.NET (.aspx) điều khiển người dùng (.ascx) và trang master page (.mater) Bạn có thể dụng các tình năng có sẵn của ASP.NET như dùng lồng các trang master page sử dụng in-line expression (<%=%>), sử dụng server controls, mẫu, data-binding, địa phương hóa (localization) và hơn thế nữa.
* Hỗ trợ các tính năng có sẵn của ASP.NET như cơ chế xác thực người dùng, quản lý thành viên, quyền, output caching và data caching, seession và profile, quản lý tình trạng ứng dụng, hệ thống cấu hình…
* ASP.NET MVC5 còn bổ sung một view engine mới là Razor View Engine cho phép thiết lập các view nhanh chóng, dễ dàng và tốn ít công sức hơn so với việc sử dụng Web Forms view engine.

2.2. Tổng quan về SQL SERVER

2.2.1. Hệ quản trị CSDL SQL Server

Khái niệm hệ quản trị cơ sở dữ liệu SQL Server

SQL Server là một hệ thống quản lý cơ sở dữ liệu RDBMS sử dụng Transact-SQL để trao đổi dữ liệu giữa Client computer và SQL Server computer. Một RDBMS bao gồm Databases, Database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS.

SQL Server được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera -Byte và có thể phục vụ cùng lúc cho hàng ngàn User. SQL Server 2000 có thể kết hợp "ăn ý" với các Server khác như Microsoft Internet Information Server (IIS), ECommerce Server, Proxy Server...

**2.2.2. Đặc điểm của SQL Server và đối tượng làm việc**

* SQL là ngôn ngữ tựa tiếng Anh.
* SQL là ngôn ngữ phi thủ tục, nó không yêu cầu ta cách thức truy nhập CSDL như thế nào. Tất cả các thông báo của SQL đều rất dễ sử dụng và ít khả năng mắc lỗi .
* SQL cung cấp tập lệnh phong phú cho các công việc hỏi đáp dữ liệu
* Chèn, cập nhật, xoá các hàng trong một quan hệ.
* Tạo, sửa đổi, thêm và xoá các đối tượng trong của CSDL.
* Điều khiển việc truy nhập tới cơ sở dữ liệu và các đối tượng của CSDL để đảm bảo tính bảo mật của cơ sở dữ liệu.
* Đảm bảo tính nhất quán và sự ràng buộc của CSDL.
* Yêu cầu duy nhất để sử dụng cho các hỏi đáp là phải nắm vững được các cấu trúc CSDL của mình.

# CHƯƠNG 3: NỘI DUNG THỰC HIỆN

## . Đối tượng sử dụng website:

* Đối với quản lý kí túc xá: Quản lý kí túc xá có quyền quản lý (thêm/sửa/xóa) các thông tin liên quan đến phòng ở (phòng, giường, tủ), sinh viên, hợp đồng thuê phòng, thu chi, thanh toán, đăng ký sử dụng dịch vụ.
* Đối với nhân viên: Nhân viên có quyền thêm thông tin liên quan đến sinh viên, hợp đồng thuê phòng, thu chi, thanh toán và đăng ký sử dụng dịch vụ.

## 3.2. Các chức năng chính của website:

Chức năng đối với Admin

1. ***Quản lý tài khoản***

* Đăng nhập vào hệ thống bằng cách cung cấp tên đăng nhập và mật khẩu.
* Kiểm tra tính hợp lệ của thông tin đăng nhập.
* Chuyển đến trang quản trị nếu thông tin đăng nhập hợp lệ, ngược lại thông báo lỗi và yêu cầu nhập lại.

1. ***Quản lí phòng và giường***

Quản trị viên có quyền thêm, sửa hoặc xóa phòng và giường.

* Thêm:
* Kiểm tra tính hợp lệ của thông tin phòng và giường.
* Kiểm tra số ID phòng và giường có tồn tại hay chưa?
* Nếu chưa thêm mới phòng hoặc giường vào CSDL.
* Sửa:
* Kiểm tra tính hợp lệ của thông tin phòng và giường.
* Kiểm tra số ID phòng và giường có tồn tại hay chưa?
* Nếu chưa cập nhật thông tin phòng hoặc giường vào CSDL.
* Xóa:
* Nếu phòng hoặc giường đang có người ở thì không được phép xóa.
* Ngược lại cho phép xóa phòng hoặc giường.

1. ***Quản lý khách hàng***

Quản trị viên có quyền thêm, sửa hoặc xóa thông tin khách hàng.

* Thêm:
* Kiểm tra tính hợp lệ của thông tin khách hàng.
* Kiểm tra số ID khách hàng có tồn tại hay chưa?
* Nếu chưa thêm mới thông tin khách hàng vào CSDL.
* Sửa:
* Kiểm tra tính hợp lệ của thông tin khách hàng.
* Kiểm tra số ID khách hàng có tồn tại hay chưa?
* Nếu chưa cập nhật thông tin khách hàng vào CSDL.
* Xóa:
* Nếu khách hàng đang thuê phòng hoặc có hóa đơn chưa thanh toán thì không được phép xóa.
* Ngược lại cho phép xóa thông tin khách hàng.

1. ***Quản lý hóa đơn***

Quản trị viên có thể xem, xóa và giải quyết hóa đơn.

* Xem:
* Hiển thị tất cả các hóa đơn, phân theo tình trạng: chưa thanh toán và đã thanh toán.
* Xóa:
* Chỉ được phép xóa các hóa đơn chưa thanh toán.
* Các hóa đơn đã thanh toán thì không được xóa.
* Thanh toán hóa đơn
* Chuyển hóa đơn chưa thanh toán sang trạng thái đã thanh toán..

3.3 Phân tích thiết kế hệ thống

3.3.1 Các yêu cầu chức năng chính của hệ thống

***a)Yêu cầu lưu trữ***

***-*** Hệ thống cần lưu trữ các thông tin liên quan đến sinh viên, phòng, hợp đồng thuê, thanh toán và các thông tin liên quan khác.

* Thông tin đăng nhập
* Phần quản trị của hệ thống cần đảm bảo tính bảo mật để truy cập tới toàn bộ thông tin của kí túc xá.
* Thông tin sinh viên (Mã sinh viên, Họ và tên, Ngày sinh, Số điện thoại, Email, Địa chỉ thường trú, Địa chỉ tạm trú, Giới tính, Quốc tịch, Số CMND/CCCD/Hộ chiếu, Ngày cấp, Nơi cấp, Ngày hết hạn, Lớp, Khoa)
* Thông tin phòng (Mã phòng, Loại phòng, Sức chứa, Giá thuê, Trạng thái)
* Thông tin hợp đồng thuê (Mã hợp đồng, Mã sinh viên, Mã phòng, Ngày thuê, Ngày trả, Giá thuê)
* Thông tin thanh toán (Mã thanh toán, Mã hợp đồng, Ngày thanh toán, Số tiền thanh toán, Phương thức thanh toán)

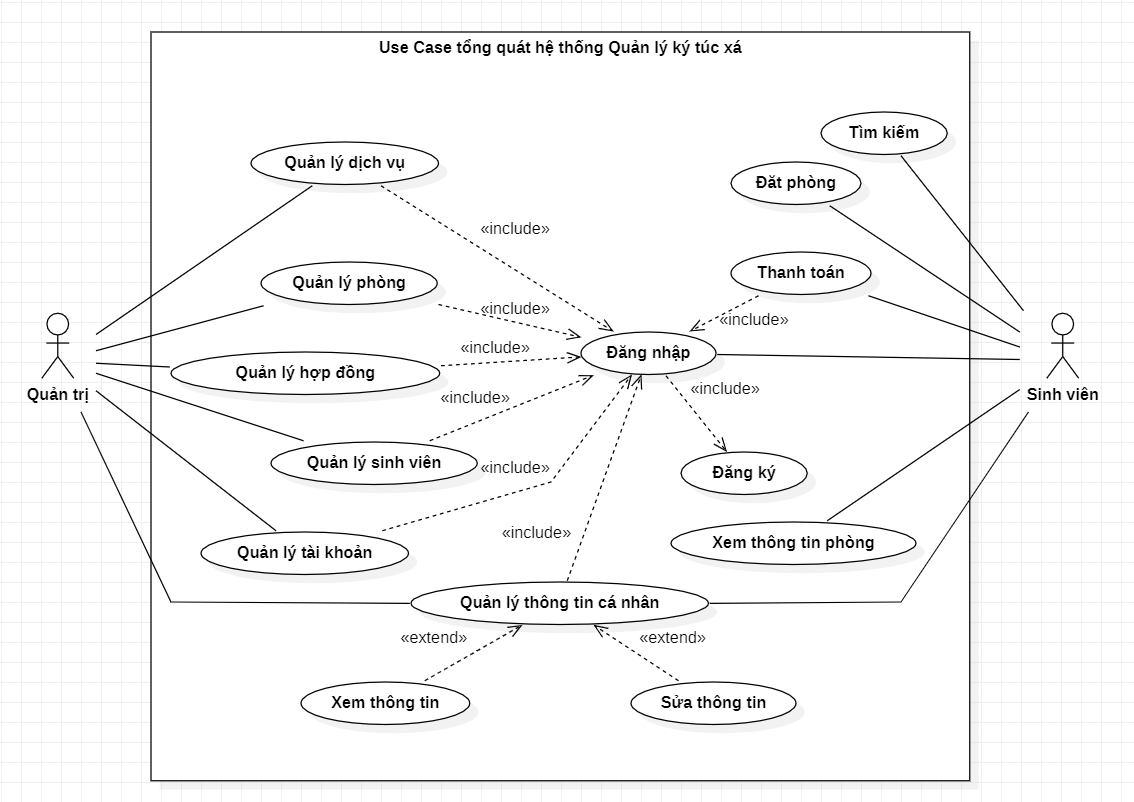
***b) Yêu cầu tính toán***

***-*** Tính tổng số tiền phải thanh toán cho mỗi hợp đồng thuê theo thời gian thuê và giá thuê của phòng.

***c)Yêu cầu tra cứu***

* Chọn thông tin tra cứu về sinh viên, phòng hoặc hợp đồng thuê.
* Kết quả xuất ra sau khi tra cứu: Danh sách các thông tin liên quan (nếu có) theo tiêu chuẩn đã được đưa ra.

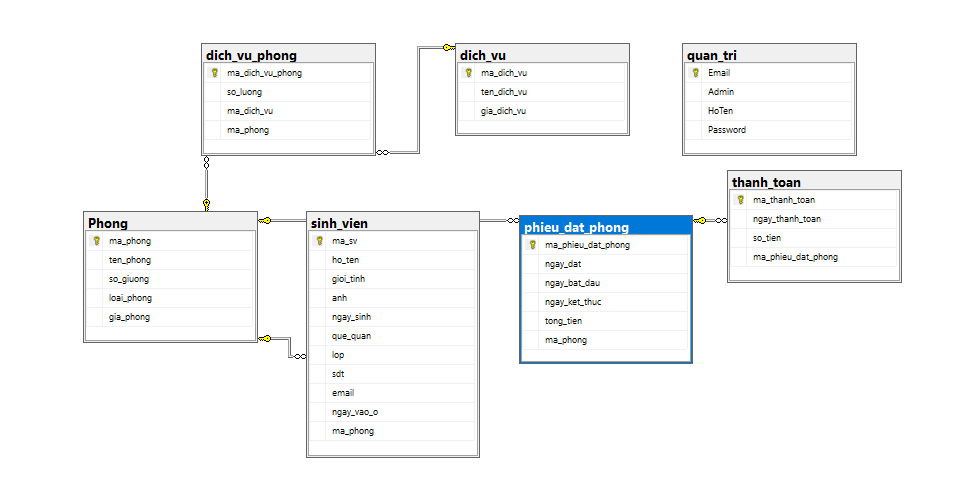
**3.3.2 Các biểu đồ Use Case**



***Hình 3: Biểu đồ Use case tổng quát***

3.4Thiết kế cơ sở dữ liệu

3.4.1 Mô hình cơ sở dữ liệu

***Hình 4: Biểu đồ dữ liệu quan hệ***

3.6.2 Danh sách các lớp đối tượng

***a. Lớp Phòng***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | ma\_phong | VARCHAR(10) | Khóa chính | Mã phòng |
| 2 | ten\_phong | NVARCHAR(50) |  | Tên phòng |
| 3 | so\_giuong | INT |  | Số giường |
| 4 | loai\_phong | NVARCHAR(50) |  | Loại phòng |
| 5 | gia\_phong | DECIMAL(10,2) |  | Giá phòng |

*Bảng 3-1: Danh sách các thuộc tính lớp Phòng*

***b. Lớp Sinh Viên***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | ma\_sv | VARCHAR(5) | Khóa chính | Mã sinh viên |
| **2** | ho\_ten | NVARCHAR(50) |  | Họ tên sinh viên |
| **3** | gioi\_tinh | VARCHAR(10) |  | Giới tính |
| **4** | ngay\_sinh | DATE |  | Ngày sinh |
| **5** | que\_quan | VARCHAR(50) |  | Quê quán |
| **6** | lop | VARCHAR(50) |  | Lớp |
| **7** | sdt | VARCHAR(20) |  | Số điện thoại |
| **8** | email | VARCHAR(50) |  | Email |
| **9** | ngay\_vao\_o | DATE |  | Ngày vào ở |
| **10** | ma\_phong | VARCHAR(10) |  | Mã phòng |

*Bảng 3-2: Danh sách các thuộc tính lớp Sinh Viên*

1. ***Lớp phiếu đặt phòng***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | ma\_phieu\_dat\_phong | VARCHAR(10) | Khóa chính | Mã phiếu đặt phòng |
| **2** | ngay\_dat | DATE |  | Ngày đặt |
| **3** | ngay\_bat\_dau | DATE |  | Ngày bắt đầu |
| **4** | ngay\_ket\_thuc | DATE |  | Ngày kết thúc |
| **5** | tong\_tien | DECIMAL(10,2) |  | Tổng tiền |
| **6** | ma\_phong | VARCHAR(10) |  | Mã phòng |

*Bảng 3-3: Danh sách các thuộc tính lớp Phiếu đặt phòng*

1. ***Lớp Thanh Toán***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | ma\_thanh\_toan | VARCHAR(10) | Khóa chính | Mã thanh toán |
| **2** | ngay\_thanh\_toan | DATE |  | Ngày thanh toán |
| **3** | so\_tien | DECIMAL(10,2) |  | Số tiền |
| **4** | ma\_phieu\_dat\_phong | VARCHAR(10) |  | Mã phiếu đặt phòng |

*Bảng 3-4: Danh sách các thuộc tính lớp Thanh toán*

1. ***Lớp Dịch Vụ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | ma\_dich\_vu | VARCHAR(10) | Khóa chính | Mã dịch vụ |
| **2** | ten\_dich\_vu | VARCHAR(50) |  | Tên dịch vụ |
| **3** | gia\_dich\_vu | DECIMAL(10,2) |  | Giá dịch vụ |

*Bảng 3-5: Danh sách các thuộc tính lớp Dịch vụ*

1. ***Lớp Dịch Vụ Phòng***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | ma\_dich\_vu\_phong | VARCHAR(10) | Khóa chính | Mã dịch vụ phòng |
| **2** | so\_luong | INT |  | Số lượng |
| **3** | ma\_dich\_vu | VARCHAR(10) |  | Mã dịch vụ |
| **4** | ma\_phong | VARCHAR(10) |  | Mã phòng |

*Bảng 3-6: Danh sách các thuộc tính lớp Dịch vụ phòng*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| **1** | Email | VARCHAR(50) | Khóa chính | Email |
| **2** | Admin | Bit |  | Admin |
| **3** | HoTen | VARCHAR(50) |  | Họ tên |
| **4** | Password | VARCHAR(50) |  | Password |

*Bảng 3-6: Danh sách các thuộc tính lớp Quản trị*

3.5 Thiết kế và đặc tả giao diện

3.5.1 Giao diện trang đăng nhập



###### ***Hình 5: Giao diện trang đăng nhập***

* Mục đích: để thực hiện việc đặt hàng (đối với khách hàng) hoặc quản lý sản phẩm (đối với quản trị viên).
* Khách hàng muốn mua hàng thì phải đăng nhập vào hệ thống bằng địa chỉ email và password đã đăng ký.
* Khi đăng nhập thành công:
* Đối với khách hàng: Tên khách hàng sẽ được hiện thị lên trên tiêu đề của trang ngược lại nếu đăng nhập sai thì hệ thống sẽ yêu cầu nhập lại.
* Đối với quản trị: Tên Admin sẽ được hiện thị lên trên tiêu đề của trang ngược lại nếu đăng nhập sai thì hệ thống sẽ yêu cầu nhập lại.
* Khi hoàn tất quá trình đăng nhập:
* Đối với khách hàng: Thì hệ thống sẽ trở lại trang chủ để cho bạn tìm kiếm sản phẩm muốn mua.
* Đối với quản trị: Thì hệ thống sẽ truy cập trực tiếp vào trang quản trị.

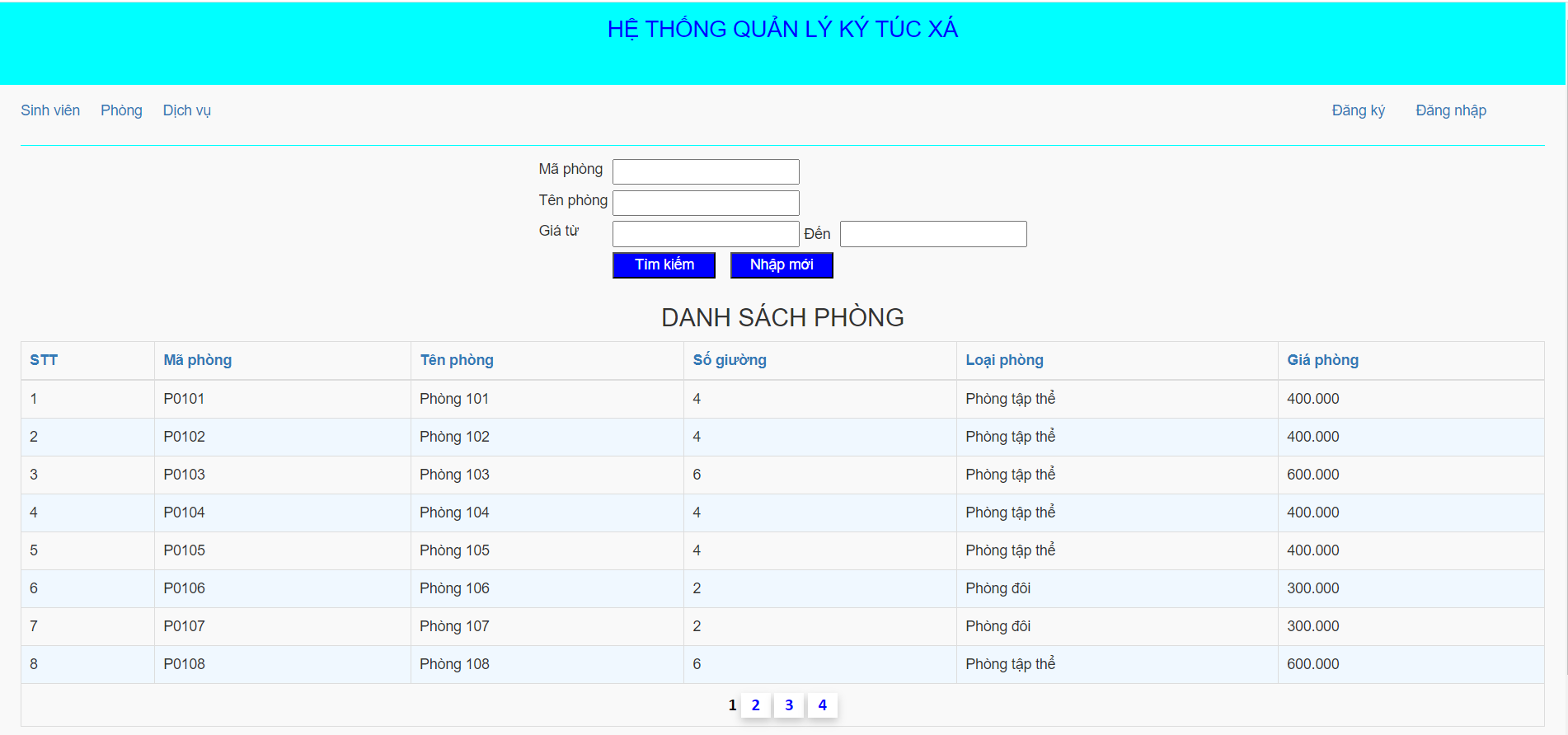
3.5.2 Giao diện trang đăng ký



###### ***Hình 6: Giao diện trang đăng ký***

* Mục đích: đăng ký thông tin để thực hiện việc mua sản phẩm tại website.
* Ràng buộc: thông tin phải chính xác.
* Tại trang đăng ký thông tin khách hàng sẽ đăng ký những thông tin mà hệ thống yêu cầu. Sau khi đăng ký xong thông tin sẽ được hệ thống xử lý khách hàng quay về trang đăng nhập để đăng nhập thực hiện mua hàng.
* Thông tin đăng ký phải đăng ký đầy đủ thông tin nếu không điền đủ thông tin thì hệ thống sẽ báo bạn phải điền đủ thông tin. Đối với email thì hệ thống có kiểm tra bạn nhập email có hợp lệ không nếu không hợp lệ yêu cầu bạn nhập lại. Khi nhập mật khẩu bạn phải nhập mật khẩu đúng nếu không trùng khớp nhau hệ thống sẽ phản hồi lại mật khẩu không hợp lệ.

3.5.3 Giao diện trang danh sách phòng ở



###### ***Hình 7: Giao diện trang danh sách phòng ở***

* Mục đích: Hiển thị thông tin của mỗi phòng trong ký túc xá.
* Để thuận lợi cho việc lựa chọn phòng phù hợp với nhu cầu.
* Sinh viên có thể tìm kiếm để tìm phòng phù hợp với bản thân

3.5.4 Giao diện trang danh sách sinh viên



*Hình 8: Giao diện trang chi trang danh sách sinh viên*

* Mục đích: Hiển thị thông tin của sinh viên có trong ký túc xá
* Có thể biết được phòng nào đã có người, phòng nào còn trống
* Sinh viên có thể tìm kiếm để tìm phòng phù hợp với bản thân

3.5.5 Giao diện dịch vụ



***Hình 10: Giao diện dịch vụ***

* Mục đích: Cho biết hiện tại trong ký túc xá có những dịch vụ gì

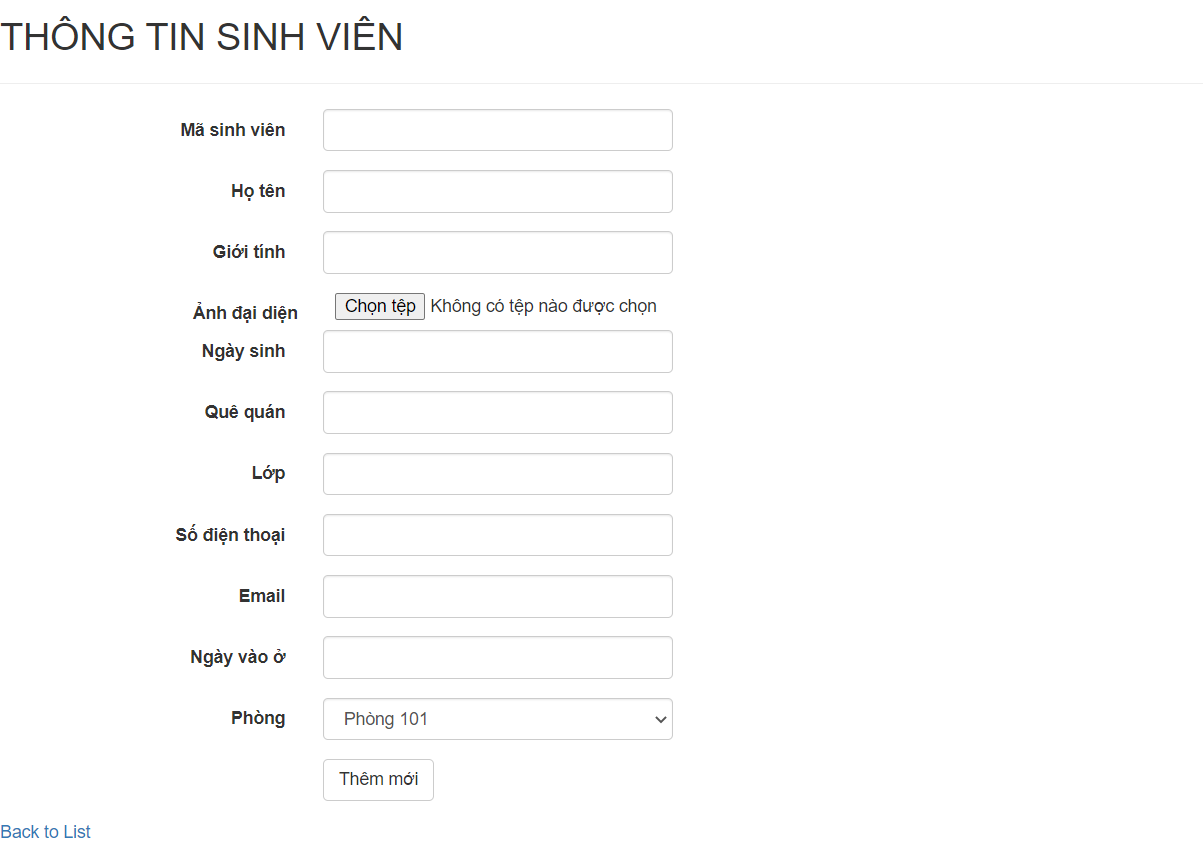
3.5.6 Giao diện trang quản lý sinh viên

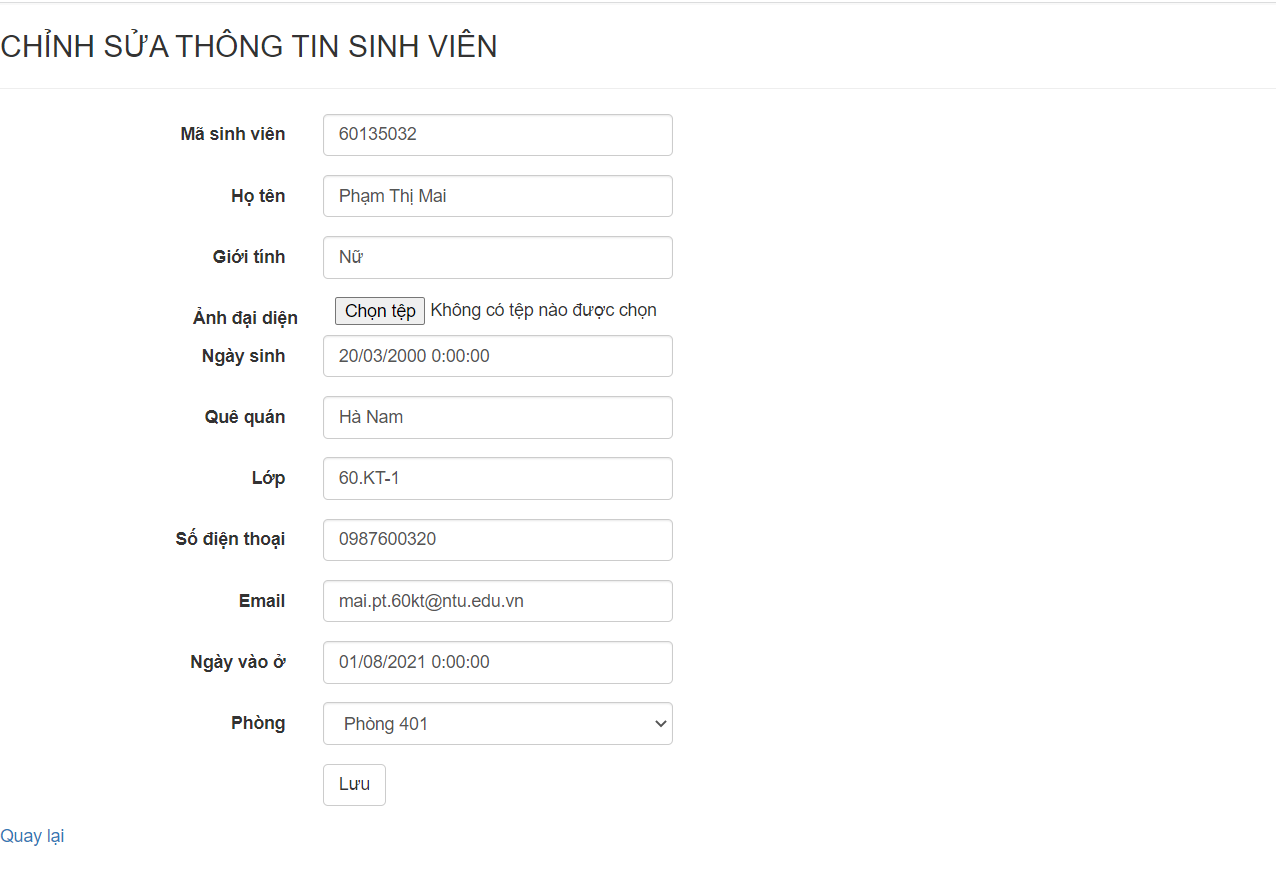


***Hình 11: Giao diện trang quản lý sinh viên***

* Mục đích: Thực hiện các thao tác như thêm, sửa, xóa, xem chi tiết sinh viên.
* Ràng buộc: Đăng nhập dưới quyền quản trị.
* Mô tả:
* Tại cột chức năng, nhấn icon đầu tiên để sửa thông tin sinh viên, nhấn icon thứ hai để xem chi tiết, nhấn icon thứ ba để xóa sinh viên.
* Để thêm hoặc tìm kiếm sinh viên thì ta có thể đặt chuột tại mục “Sinh viên” ở thanh tiêu đề để hiện danh sách chức năng ở đó.

3.5.7 Giao diện trang thêm, sửa, xóa sinh viên

  
***Hình 12: Giao diện trang thêm sinh viên***



***Hình 13: Giao diện trang sửa thông tin sinh viên***



***Hình 14: Giao diện trang xóa thông tin sinh viên***

* Mục đích: thực hiện việc thêm, sửa, xóa sinh viên
* Ràng buộc: đăng nhập dưới quyền của người quản trị.
* Mô tả:
* Thêm: Thực hiện thêm một sinh viên mới vào danh sách.
* Sửa: Sửa thông tin sinh viên.
* Xóa: Xóa thông tin sinh viên.

3.5.8 Giao diện trang quản lý phòng ở



***Hình 15: Giao diện trang quản lý phòng ở***

* Mục đích: thực hiện nhập, sửa, xóa thông tin phòng ở.
* Ràng buộc: đăng nhập dưới quyền là người quản trị.
* Mô tả:
* Tại cột chức năng, nhấn icon đầu tiên để sửa thông tin phòng, nhấn icon thứ hai để xem chi tiết, nhấn icon thứ ba để xóa phòng.
* Để thêm hoặc tìm kiếm phòng thì ta có thể đặt chuột tại mục “Phòng” ở thanh tiêu đề để hiện danh sách chức năng ở đó.

3.5.9 Giao diện trang quản lý dịch vụ



***Hình 16: Giao diện trang quản lý dịch vụ***

* Mục đích: thực hiện nhập, sửa, xóa thông tin dịch vụ.
* Ràng buộc: đăng nhập dưới quyền là người quản trị.
* Mô tả:
* Tại cột chức năng, nhấn icon đầu tiên để sửa thông tin dịch vụ, nhấn icon thứ hai để xem chi tiết, nhấn icon thứ ba để xóa.
* Để thêm dịch vụ thì ta có thể đặt chuột tại mục “Dịch vụ” ở thanh tiêu đề để hiện danh sách chức năng ở đó.

3.5.10 Giao diện trang quản lý dịch vụ phòng

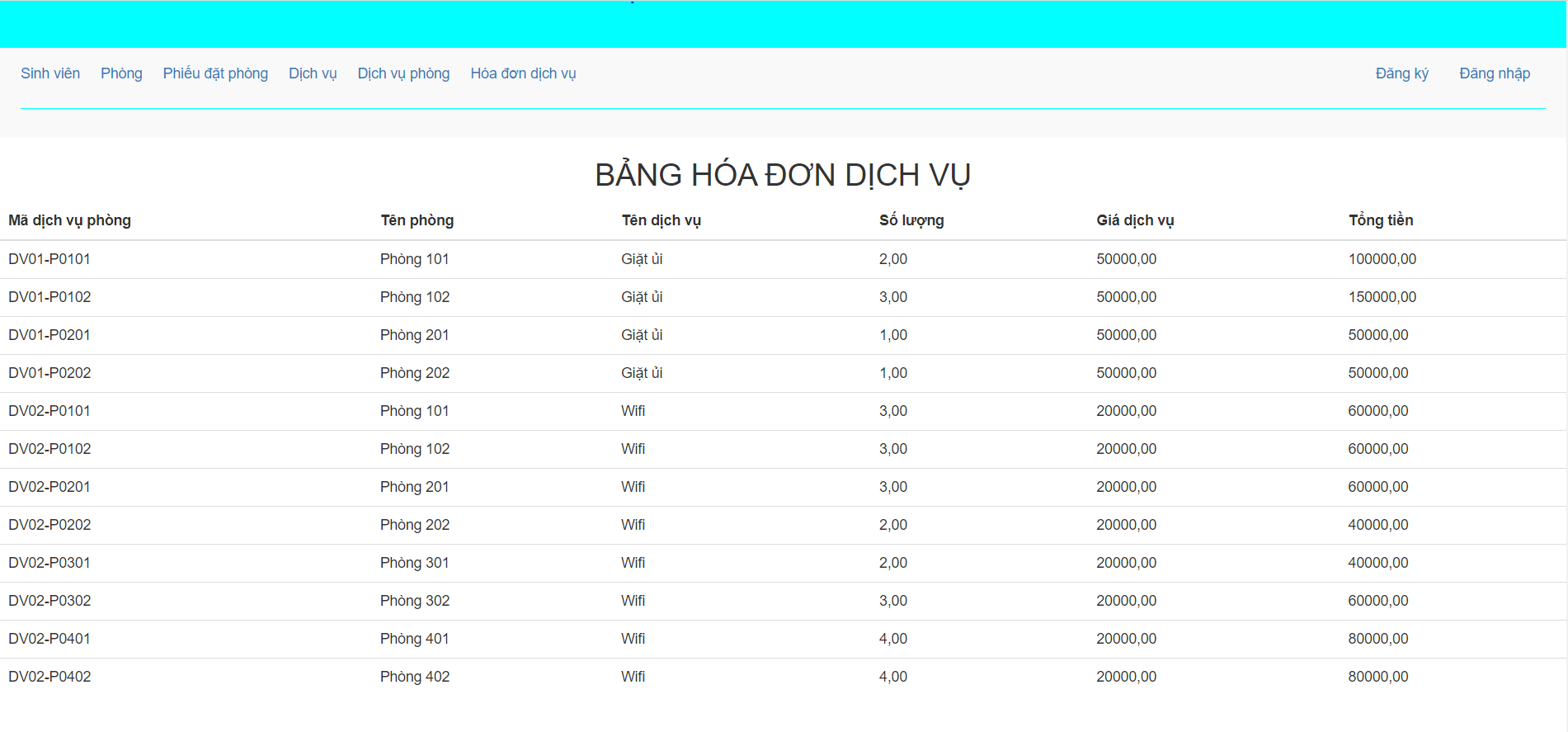


***Hình 17: Giao diện trang quản lý dịch vụ phòng***

* Mục đích: thực hiện nhập, sửa, xóa thông tin dịch vụ phòng.
* Ràng buộc: đăng nhập dưới quyền là người quản trị.
* Mô tả:
* Tại cột chức năng, nhấn icon đầu tiên để sửa thông tin dịch vụ phòng, nhấn icon thứ hai để xem chi tiết, nhấn icon thứ ba để xóa.
* Để thêm dịch vụ phòng thì ta có thể đặt chuột tại mục “Dịch vụ phòng” ở thanh tiêu đề để hiện danh sách chức năng ở đó.

3.5.10 Giao diện trang quản lý hóa đơn dịch vụ

Mục đích: Tính tiền dịch vụ ở mỗi phòng.



***Hình 18: Giao diện trang quản lý hóa đơn dịch vụ***

# CHƯƠNG 4: KẾT LUẬN

4.1 Kết quả đạt được của đề tài

* Xây dựng thành công website Quản lý kí túc xá.
* Giao diện website được thiết kế đẹp, hài hòa về màu sắc, dễ nhìn và dễ sử dụng.
* Xây dựng và triển khai các chức năng cần thiết cho hệ thống, bao gồm thêm, sửa, xóa thông tin.

4.2 Hạn chế của đề tài

* Một số chức năng vẫn chưa được tối ưu hoặc triển khai một cách hoàn thiện.
* Chưa có chức năng quản lý hội viên sử dụng kí túc xá.
* Còn một số yếu điểm khác...

4.3 Kết luận

Trong quá trình nghiên cứu và áp dụng kiến thức đã học, chúng em đã hoàn thành thành công phần mềm "Quản lý kí túc xá" nhằm mục đích cung cấp một hệ thống quản lý hiệu quả cho kí túc xá và củng cố lại kiến thức đã học. Chúng em hy vọng rằng qua hoạt động nhóm này, chúng tôi đã tạo nền tảng để phát triển kiến thức và nâng cao chất lượng phần mềm.

Trong quá trình hoàn thiện phần mềm, dù chúng em có hạn chế về kinh nghiệm thực tế, báo cáo vẫn không tránh khỏi các sai sót. Chúng em rất mong nhận được sự góp ý từ quý thầy/cô để hoàn thiện hơn.

Chân thành cảm ơn quý thầy/cô đã đồng hành và hỗ trợ chúng em trong quá trình thực hiện đề tài này.

**TÀI LIỆU THAM KHẢO**

* Phạm Hữu Khang, Lập trình ASP.Net MVC. Hà Nội, Việt Nam: Nhà xuất bản Lao Động Xã Hội, 2005.
* Bùi Chí Thành , *Giáo trình Phát triển ứng dụng web.*
* Đoàn Thiện Ngân, Phạm Hữu Khang, Lập trình ASP.NET MVC , Nhà xuất bản lao động xã hội, 2004.
* Phạm Hữu Khanh, [Microsoft SQL Server 2008 - Quản Trị Cơ Sở Dữ Liệu](https://www.vinabook.com/microsoft-sql-server-2008-quan-tri-co-so-du-lieu-tap-1-p37708.html), NXB Lao động - Xã hội,2004.