TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO BÀI TẬP MÔN LẬP TRÌNH NÂNG CAO

XÂY DỰNG ỨNG DỤNG
QUẢN LÝ ĐIỂM SINH VIÊN

GVHD
TS. Lê Thành Sách

Nhóm 6:
Diệp Hưng (1570209)
Biện Lê Anh Hưng (7140235)
Nguyễn Thanh Hải (7140230)

Tp. Hồ Chí Minh, Tháng 12/2015

MỤC LỤC

1. Yêu cầu bài tập

Xây dựng lược đồ quản lý điểm sinh viên sử dụng công cụ PowerDesigner: Conceptual Data Model (CDM), từ đó tiếp tục sinh ra Logical Data Model (LDM), Physical Data Model (PDM) và tạo Cơ sở dữ liệu cho ứng dụng.

Hiện thực ứng dụng quản lý điểm sinh viên với yêu cầu: cho phép truy xuất điểm của sinh viên và các thông tin khác có liên quan.

2. Phạm vi đề tài

Mỗi năm, trường đại học tiếp nhận sinh viên và trong quá trình học tập nhà trường sẽ quản lý các kết quả học tập của từng sinh viên. Trong trường có nhiều khoa ngành khác nhau, mỗi khoa có nhiều bộ môn và nhân sự của bộ môn.

Mỗi khoa cũng tổ chức các khóa học hàng năm với chương trình học của từng khóa. Chương trình học bao gồm các môn học do bộ môn quản lý.

Các bộ môn quản lý các lớp quản lý với sinh viên đã được phân vào các lớp quản lý này cùng với giáo viên chủ nhiệm của lớp.

Các môn học được tổ chức trong các học kỳ, theo các lớp môn học. Mỗi lớp như thế sẽ bao gồm các thông tin: môn học, học kỳ, phòng học, phòng kiểm tra giữa kỳ, phòng thi, bảng điểm của sinh viên tham gia học.

Các phòng học nằm trong các tòa nhà của trường.

Ngoài ra, sinh viên chỉ có thể đăng ký môn học trong chương trình học của mình.

Điểm của sinh viên liên quan đến từng môn học bao gồm: điểm giữa kỳ, điểm thi và điểm tổng kết. Bên cạnh đó, sinh viên có thể có thêm điểm thưởng bởi các hoạt động ngoại khóa và lịch sử của các điểm phúc khảo sẽ được ghi nhận nếu có, bao gồm cả ngày chấm và người chấm điểm.

Như vậy, để có được một ứng dụng quản lý điểm sinh viên hoàn chỉnh có thể triển khai ứng dụng trong thực tế đòi hỏi rất nhiều tài nguyên và công sức. Nhận thấy với thời gian hạn hẹp của môn học cùng với nhân sự 3 người của nhóm, nhóm sẽ xây dựng một ứng dụng quản lý điểm sinh viên đơn giản. Nhóm đề ra mục đích chính là nghiên cứu môn học, ứng dụng được những kiến thức đã lĩnh hội được trên lớp vào việc xây dựng ứng dụng nhỏ này.

3. Phân tích các chức năng của ứng dụng

Quản lý các môn học của các lớp theo các học kỳ và kết quả học tập của sinh viên đối với các môn học đó. Tổng kết kết quả học tập theo kỳ, theo năm, theo khóa. Danh sách tất cả các điểm có liên quan đến một môn học cụ thể, bao gồm cả lịch sử điểm phúc khảo.

Chức năng khác: cập nhật các loại danh mục dữ liệu (danh mục lớp, danh mục môn học, nhân viên, sinh viên, quản lý phòng học. . .)

3.1. Chức năng người dùng

Người dùng là sinh viên, giáo viên. . . là những người có nhu cầu xem thông tin điểm của các sinh viên, hoặc các thông tin khác về lớp quản lý. Họ chỉ có quyền xem điểm sau khi đăng nhập vào hệ thống.

Trong ứng dụng này sẽ xây dựng 2 phân quyền cho người dùng:

- Giáo viên
- Sinh viên

3.2. Chức năng quản trị

Có hai nhóm với vai trò: quản lý viên, quản trị viên. Họ phải đăng nhập vào hệ thống để sử dụng chức năng quản trị.

Quản lý viên (manager) có các chức năng:

- Như người dùng bình thường
- Quyền tạo, thay đổi, xóa thông tin các sinh viên
- Quyền tạo, thay đổi, xóa điểm của các sinh viên
- Quyền tạo, thay đổi khóa học
- Quyền tạo, thay đổi học kỳ
- Quyền tạo, thay đổi lớp quản lý
- Quyền tạo, thay đổi lớp môn học
- Quyền tạo, thay đổi, xóa môn học

Quản trị viên (administrator) có các chức năng:

- Tất cả các quyền của quản lý viên
- Quyền quản lý nhân sự
- Quyền quản lý hệ thống

4. Phân tích yêu cầu nghiệp vụ

4.1. Các thừa tác viên nghiệp vụ

Sinh viên: là khách nói chung, là những người có nhu cầu xem thông tin điểm của các sinh viên. Họ chỉ có quyền đăng ký môn học, xem điểm.

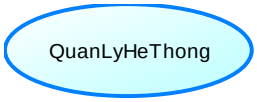
Giáo viên: có quyền xem danh sách các lớp mà họ phụ trách dạy và danh sách sinh viên trong từng lớp đó.

Quản trị viên: có tất cả các quyền của hệ thống (bao gồm cả khách và quản lý viên), nhóm này còn có thêm các chức năng quản lý người dùng, quản lý hệ thống.

Quản lý viên: là các giáo viên và nhân viên giáo vụ khoa, có tất cả các quyền của khách. Nhóm này, tùy được hệ thống thiết lập quyền mà có thêm các chức năng: quản lý môn học, quản lý điểm thi, quản lý sinh viên . . . như mô tả ở phần 3.

4.2. Một số usecase nghiệp vụ

- Usecase QuanLyHeThong

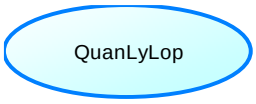


Nghiệp vụ quản lý hệ thống dành cho quản trị viên, cấu hình hệ thống, quản lý tài khoản người dùng

- Usecase QuanLySinhVien

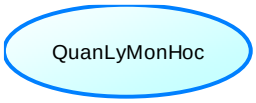
Nghiệp vụ quản lý sinh viên cho phép nhân viên quản lý cập nhật thông tin của sinh viên như thêm sinh viên mới, xóa thông tin sinh viên, sửa thông tin sinh viên.

- Usecase QuanLyLop



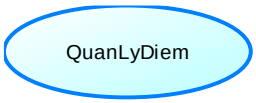
Nghiệp vụ quản lý lớp học cho phép nhân viên quản lý cập nhật thông tin của lớp quản lý và lớp môn học như thêm lớp mới, xóa thông tin lớp, sửa thông tin lớp.

- Usecase QuanLyMonHoc



Nghiệp vụ quản lý môn học cho phép nhân viên quản lý cập nhật thông tin của môn học như thêm môn học, xóa thông tin môn học, sửa thông tin môn học.

- Usecase QuanLyDiem

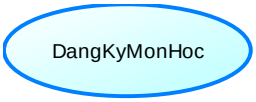


Nghiệp vụ quản lý điểm cho phép nhân viên quản lý cập nhập điểm cho sinh viên như điểm kiểm tra, điểm thi, sửa điểm cho sinh viên.

- Usecase XemDiem

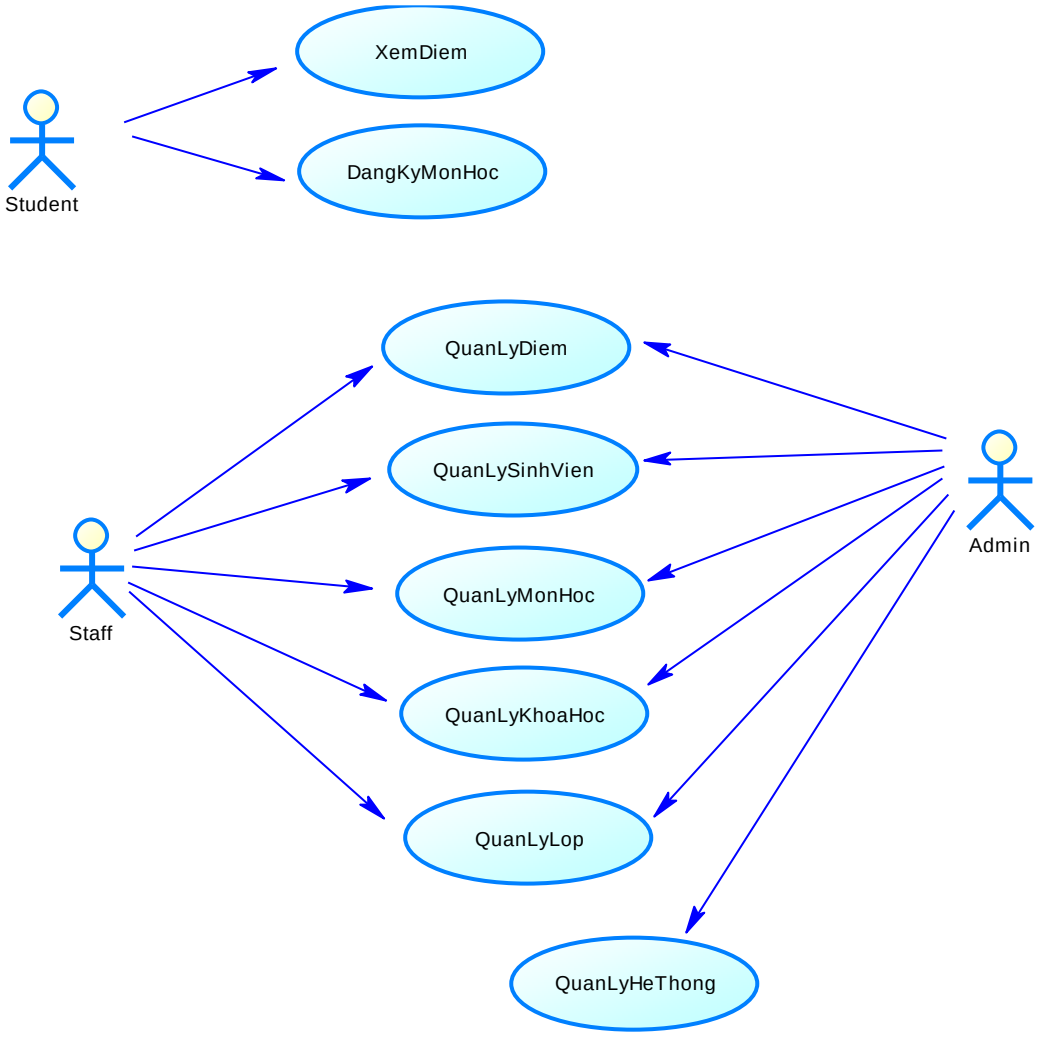
Nghiệp vụ xem điểm cho phép sinh viên truy cập vào hệ thống để xem kết quả học tập của mình

- Usecase DangKyMonHoc

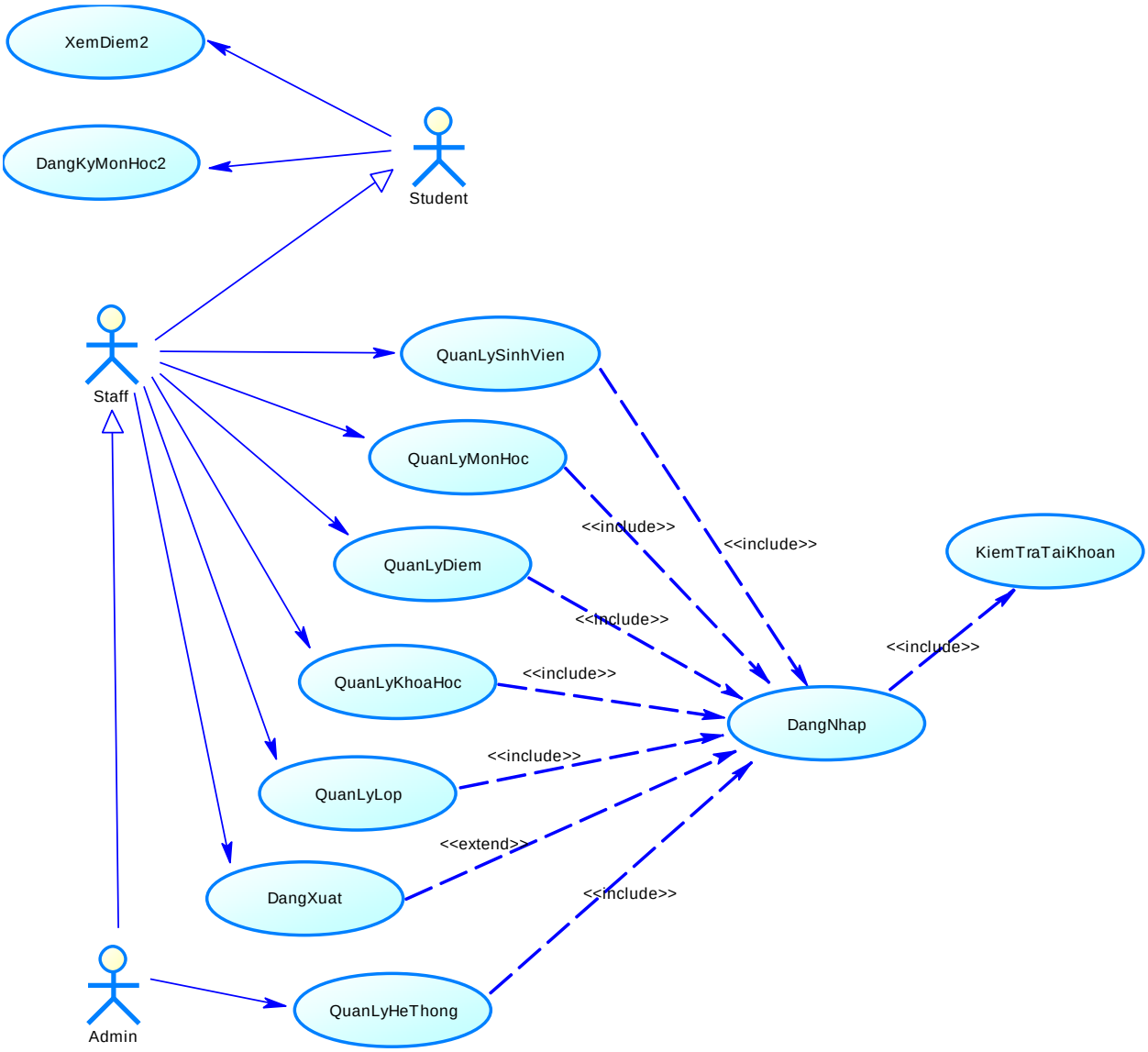


Nghiệp vụ đăng ký môn học cho phép sinh viên có thể đăng ký các môn học trong chương trình học tập.

4.3. Lược đồ usecase nghiệp vụ



4.4. Lược đồ usecase tổng quát



5. Hiện thực ứng dụng

5.1. Xây dựng Lược đồ CDM

Lược đồ CDM được xây dựng gồm các khối chính: Khoa (*Faculty*), Bộ môn (*Department*), Khóa học (*Course*), Chương trình học (*Curriculum*), Môn học (*Subject*), nhân viên (*Staff*) của trường gồm giảng viên và giáo vụ viên, Sinh viên (*Student*).

Mỗi năm học, khoa xây dựng các khóa học và các chương trình học có liên quan đến khóa học. Sinh viên sẽ được tổ chức thành các lớp quản lý thuộc các khóa học khác nhau. Nhờ vậy khi đăng ký môn học, sinh viên chỉ có thể đăng ký các môn học nằm trong chương

trình đào tạo của mình. Thông tin về điểm các môn học của sinh viên sẽ được chứa trong bảng đăng ký môn học (ScoreRecord).

Mỗi khoa sẽ bao gồm nhiều bộ môn, bộ môn gồm nhiều nhân viên là giảng viên hoặc giáo vụ viên, thực hiện công việc giảng dạy hoặc quản lý. Mỗi giảng viên sẽ phụ trách một hay nhiều lớp quản lý (*StudentClass*) cũng như các lớp môn học (*SubjectClass*) theo chuyên môn nghiệp vụ của mình.

Ngoài ra, trong lược đồ thể hiện lớp quản lý hệ thống để quản lý tài khoản: Tài khoản người dùng (*Account*), quyền (*Role*) và phân quyền (*Permission*).

Lược đồ tổng thể CDM thể hiện trong phụ lục 1 đính kèm

5.2. Tìm hiểu công nghệ nền tảng

Nhóm thực hiện ứng dụng với mục tiêu: áp dụng các kiến thức đã học trên lớp, tìm hiểu các công nghệ sẵn có, nắm bắt và áp dụng. Nhóm chọn thực hiện ứng dụng của mình trên nên web với các công nghệ nền tảng như:ASP.NET MVC, Entity Framework, Bootstrap, Jquery UI, MS SQL.

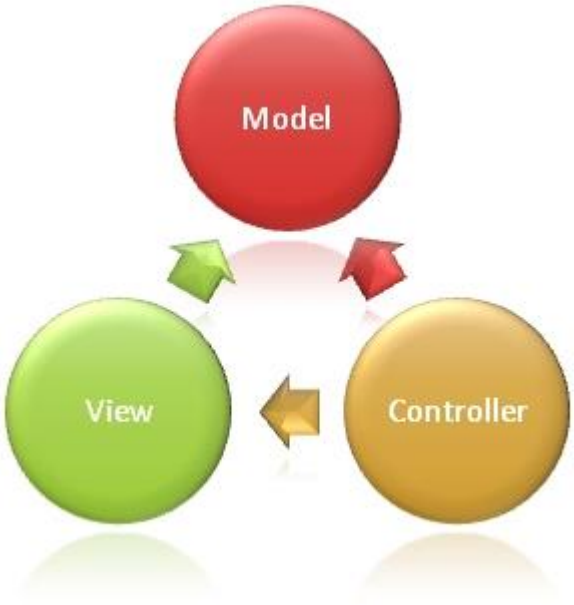
Các thành viên trong nhóm đều không chuyên về xây dựng phần mềm, đặc biệt là trên nên tảng web. Vì thế nhân cơ hội lần này nhóm cũng muồn tìm hiểu thêm kiến thức mới. Sau đây là một số giới thiệu sơ lượt mà nhóm đúc kết lại sau quá trình tìm hiểu.

5.2.1. Giới thiệu ASP.NET MVC

Mẫu kiến trúc Model – View – Controller được sử dụng nhằm chia 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.

Nền tảng MVC bao gồm các thành phần dưới đây:



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: 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 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 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 chỉ cần tập trung vào giao diện (views) mà không phải quan tâm đến logic xử lý thông tin 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.

Các tính năng của nền tảng 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 (.marter). Bạn có thể sử dụng các tính năng có sẵn của ASP.NET như là sử dụng 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…

• Từ ASP.NET MVC 3, mô hình 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.

5.2.2. Giới thiệu Entity Framework

Microsoft tạo ra "Entity Framework" để tự động hóa các hoạt động cơ sở dữ liệu liên quan tới ứng dụng. Nói cách khác Entity Framework giúp tiệp cận cơ sở dữ liệu dễ dàng và đỡ tốn công sức hơn.

Entity Framework là một bộ ánh xạ đối tượng – quan hệ cho phép người lập trình .NET làm việc với dữ liệu quan hệ qua các đối tượng (*object*), nó giúp lập trình viên không cần viết mã cho hầu hết những gì liên quan đến truy cập dữ liệu.

Có 3 cách sử dụng Entity Framework: Code First, Models First, Database First:

• *Database first*: là phương pháp chỉ nên dùng khi bạn đã có sẵn CSDL, EF Wizard sẽ tạo Model và Code.

• *Model first*: nên dùng khi bạn bắt đầu thiết kế CSDL từ đầu (từ chưa có gì). Bạn sẽ thiết kế mô hình CSDL (Model) EF sẽ tự tạo code cho bạn, sau đó nhờ EF Wizard tạo CSDL.

• *Code first*: nên dùng khi đã có mô hình CSDL, bạn sẽ viết Class, từ đó tạo Database.

Trong ứng dụng này nhóm sử dụng kết hợp giữa *Database first* và *Code first*. Tức là nhóm sử dụng công cụ Power Designer thiết kế mô hình cơ sở dữ liệu rồi đổ trực tiếp xuống database (*Database first*). Sau đó dùng công cụ *EntityFramework Reverse POCO Generator* để chuyển từ database thành các class mã nguồn để sử dụng như là *Code first*.

5.2.3. Giới thiệu Web UI Framwork

Hiện tại có rất nhiều Web UI Framework (css, js, html) để người lập trình viên sử dụng. Trong ứng dụng này nhóm sử dụng:

- Bootstrap
- Jquery UI

Hai khung giao diện web này khá phổ biến, phát triển rất mạnh trong thời gian qua. Ưu điểm chính của chúng là dễ sử dụng, khá đầy đủ chức năng và nhất là mã nguồn mở giúp lập trình viên có nhiều tùy chỉnh để phù hợp với ứng dụng của mình.

5.3. Xây dựng mã nguồn chương trình

Chương trình được tổ chức theo mô hình ASP.NET MVC chuẩn. Tuy nhiên có một số tùy chỉnh để phù hợp với ứng dụng hiện tại.

Các namespace chính của chương trình:

• EduMSDemo.Objects: Model trong mô hình MVC

• EduMSDemo.Controllers: Controller trong mô hình MVC

• EduMSDemo.Web: chứa View trong mô hình MVC và các tài nguyên của ứng dụng.

• EduMSDemo.Data: làm nhiệm vụ cấu nối, kết nối ứng dụng với CSDL

• EduMSDemo.Components: chứa các thư viện plug-in hỗ trợ

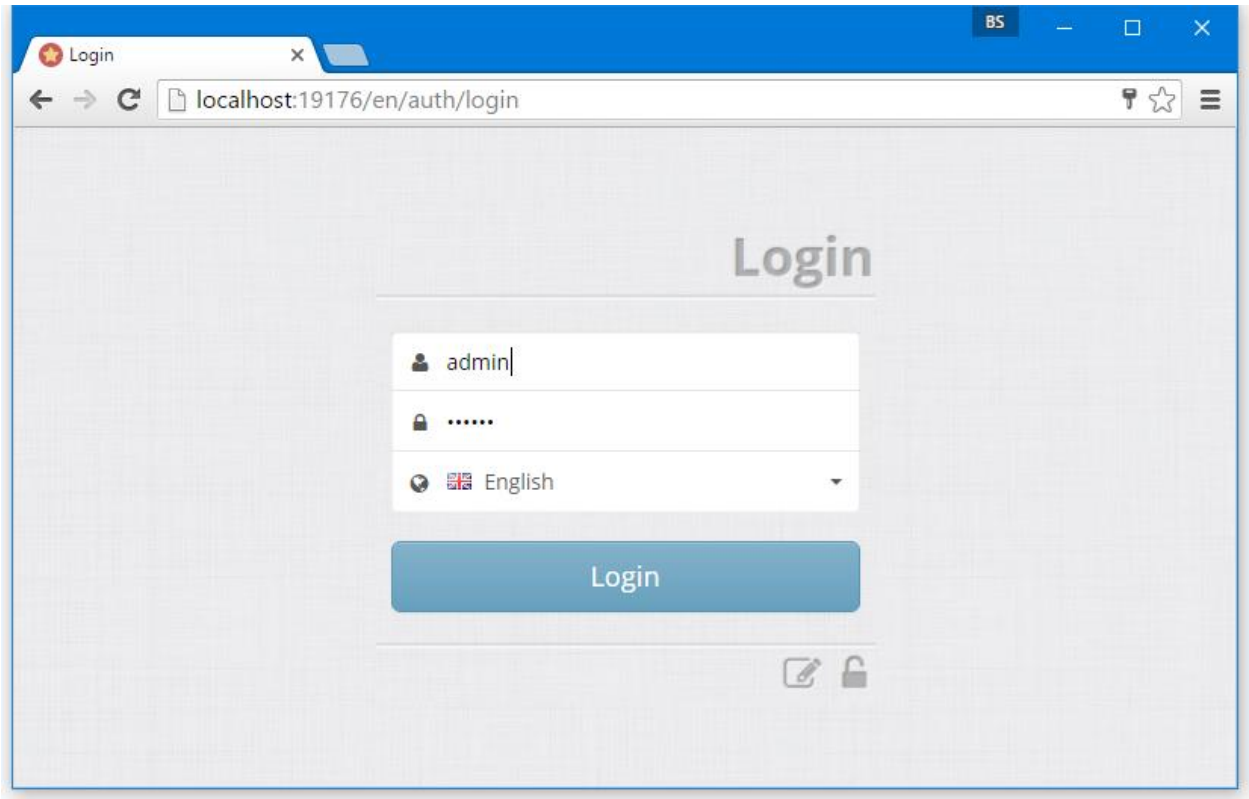
• EduMSDemo.Resources: chứa gói hỗ trợ đa ngôn ngữ

• EduMSDemo.Services: cung cấp các API để controller truy xuất vào dữ liệu

• EduMSDemo.Validators: thực hiện việc kiểm tra tính hợp lệ của dữ liệu

5.4. Xây dựng giao diện

5.4.1. Giao diện đăng nhập

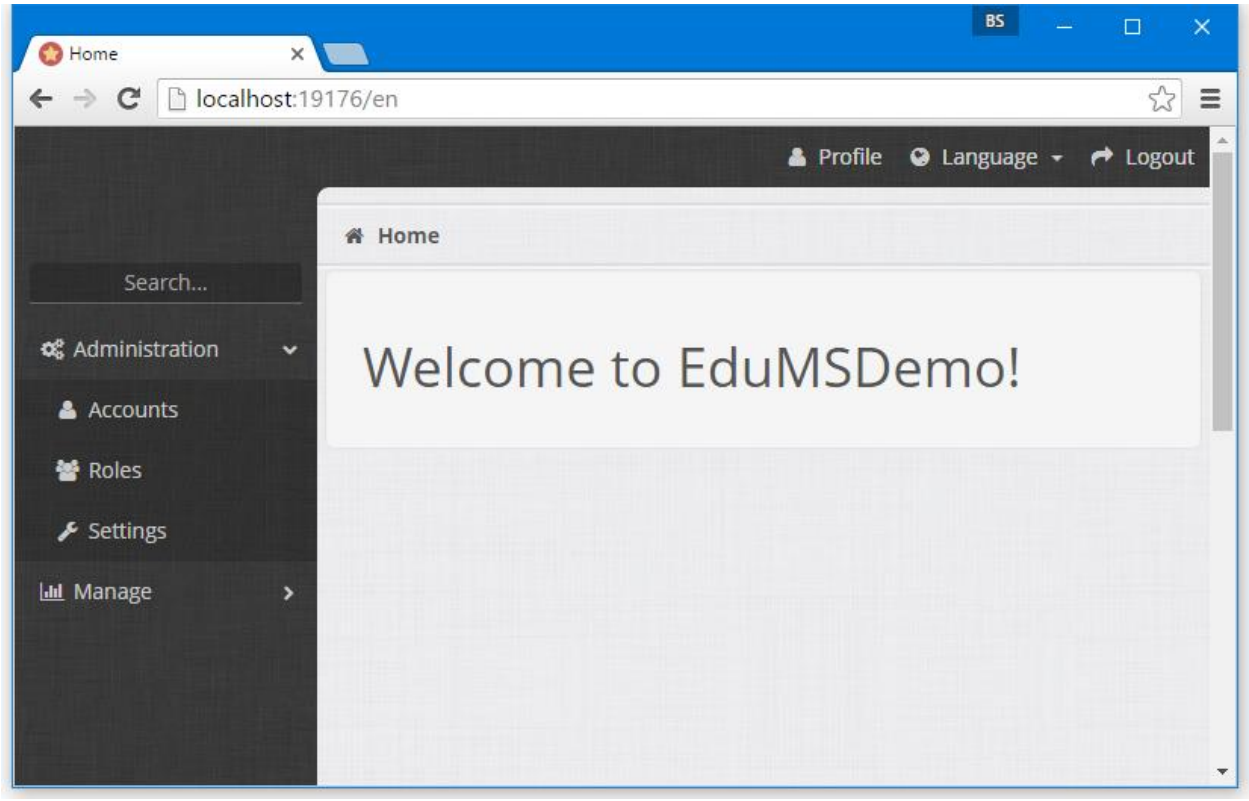


5.4.2. Giao diện trang chính

Khi đăng nhập thành công người dùng sẽ vào được trang chủ.

Các thành phần chính:

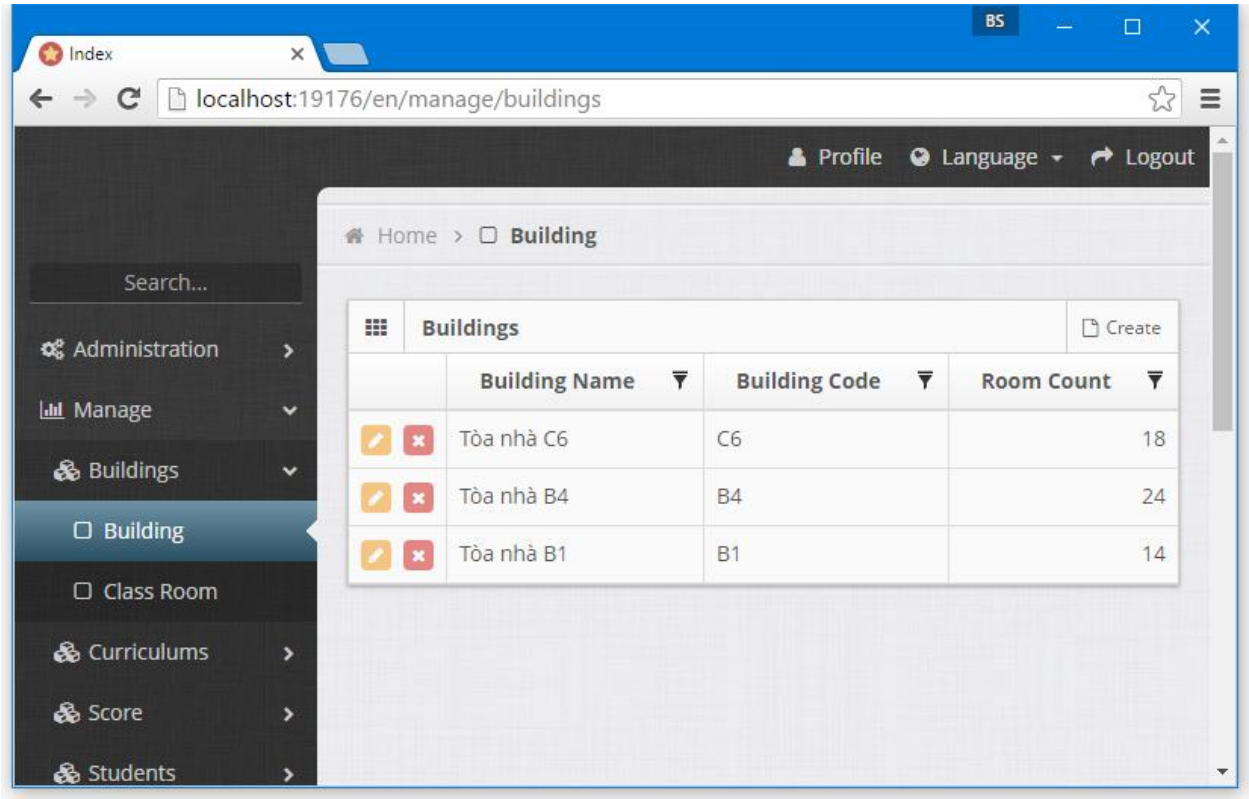
- Top menu: chứa các nút
  - Profile: để xem thông tin cá nhân
  - Language: thay đổi ngôn ngữ hiển thị
  - Logout: đăng xuất khỏi tài khoản hiện tại
- Left menu: chứa liên kết đến các trang chức năng. Tùy vào quyền hạn của tài khoản đăng nhập mà Left menu này sẽ hiển thị khác nhau.
- Main content: chính là phần nội dung của trang.



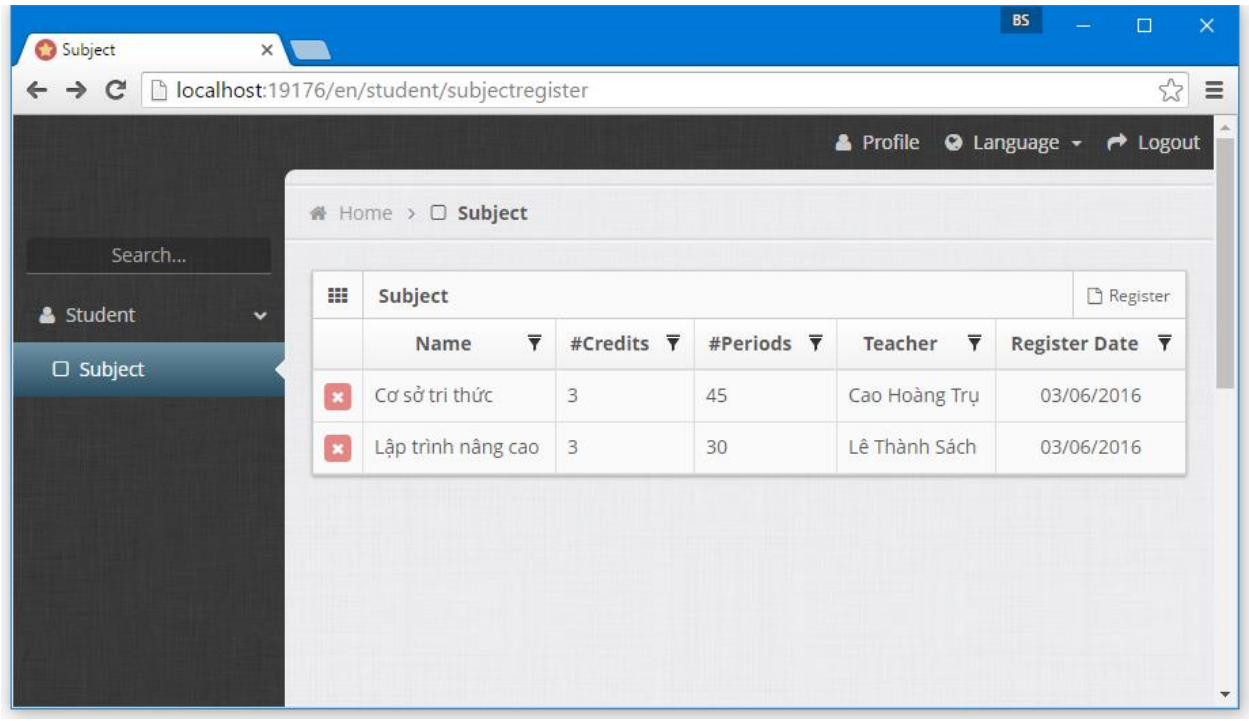
5.4.3. Giao diện các trang quản lý

Đối với việc quản lý danh sách từng loại đối tượng, ứng dụng phải có các chức năng cơ bản như:

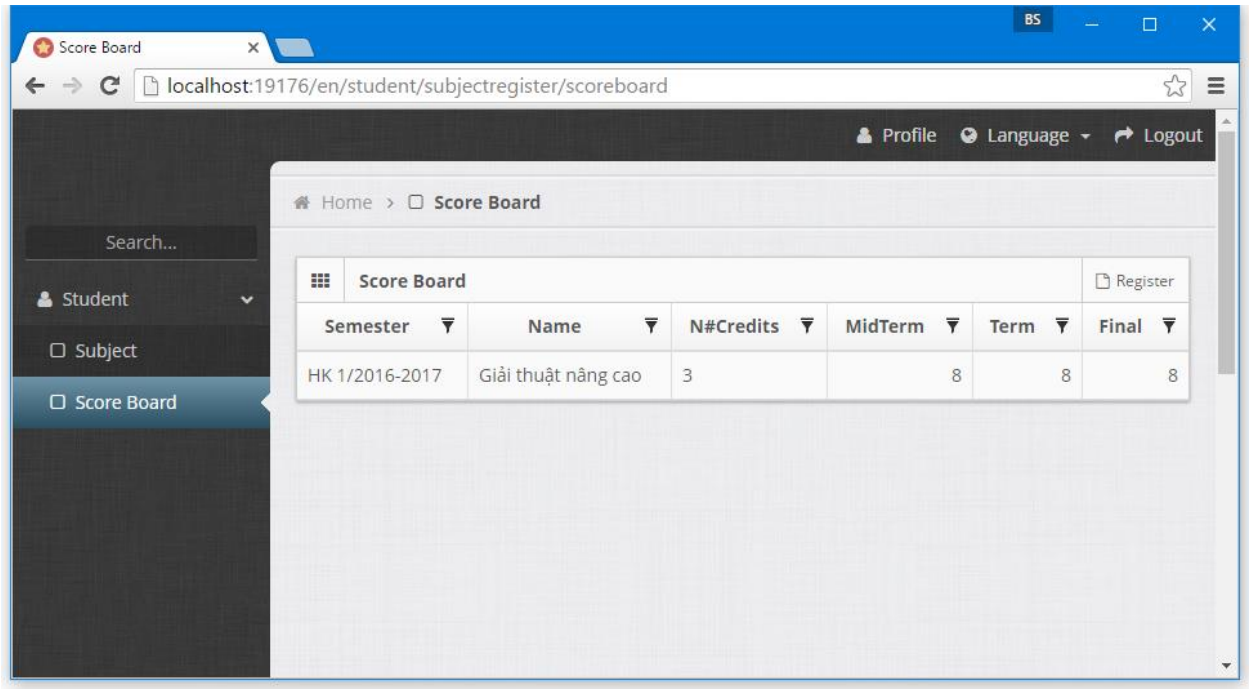
- Hiển thị danh sách
- Phân trang
- Tìm kiếm, lọc thông tin
- Thêm/Sửa/Xóa



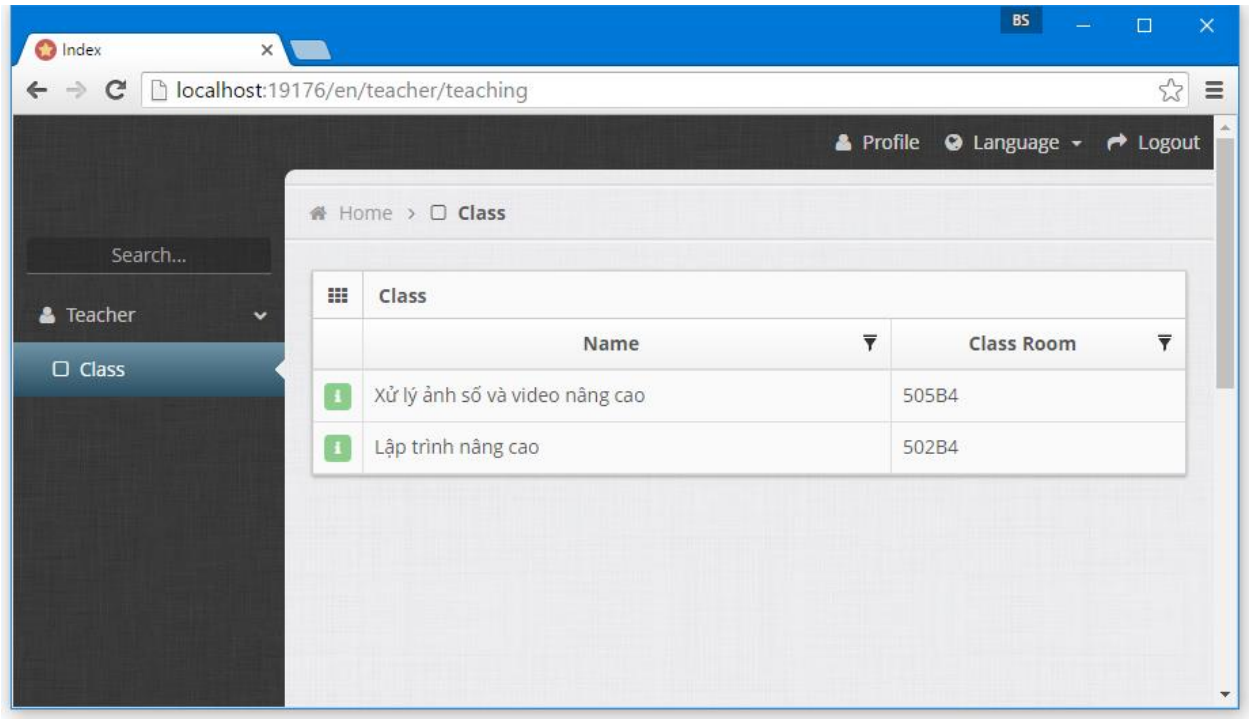
5.4.4. Giao diện trang đăng ký môn học của sinh viên



5.4.5. Giao diện trang xem điểm của sinh viên



5.4.6. Giao diện trang danh sách lớp dạy cho giáo viên



Ngoài ra còn các giao diện phụ Thêm/Sửa/Xóa được thiết kế để phù hợp với ứng dụng.

6. Tổng kết

Qua thực hiện ứng dụng quản lý điểm sinh viên bằng mô hình ASP.NET MVC, nhóm hiểu được quy trình xây dựng phần mềm từ lược đồ CDM cho đến hiện thực giao diện ứng dụng cuối cùng.

Hiện tại trong khuôn khổ bài tập ứng dụng, nhóm chỉ mới thực hiện một số chức năng cơ bản của hệ thống như đăng nhập, quản lý điểm, xem điểm sinh viên theo mô hình đã tạo. Để có thể trở thành một ứng dụng hoàn chỉnh, cần phải phân tích cụ thể hơn và xây dựng mã nguồn hoàn chỉnh hơn.

Tham khảo

[1] *EntityFrameworkTutorial*, http://www.entityframeworktutorial.net/

[2] Microsoft, *MSDN*, https://msdn.microsoft.com/en-us/

[3] Muchiachio, *ASP.NET MVC.Template introduction*,

http://www.codeproject.com/Articles/820836/ASP-NET-MVC-Template-introduction

Phụ lục 1

Lược đồ cơ sở dữ liệu