**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**LƯƠNG QUỐC KHANG**

**KHÓA LUẬN TỐT NGHIỆP**

**ỨNG DỤNG HỖ TRỢ THI TOÁN TRÁC NGHIỆM THPT ONLINE**

**MULTICHOICE QUESTION ONLINE MATH TEST**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2020**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**LƯƠNG QUỐC KHANG – 15520342**

**KHÓA LUẬN TỐT NGHIỆP**

**ỨNG DỤNG HỖ TRỢ THI TOÁN TRÁC NGHIỆM THPT ONLINE**

**MULTICHOICE QUESTION ONLINE MATH TEST**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**PGS.TS ĐỖ VĂN NHƠN**

**TP. HỒ CHÍ MINH, 2020**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

|  |  |
| --- | --- |
| **ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | **TP. HCM, ngày…..tháng…..năm……..** |

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**

**(CỦA CÁN BỘ HƯỚNG DẪN)**

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **ỨNG DỤNG HỖ TRỢ THI TOÁN TRÁC NGHIỆM THPT ONLINE** | | |
| Nhóm SV thực hiện: | | Cán bộ hướng dẫn: |
| Lương Quốc Khang | 15520342 | PGS. TS Đỗ Văn Nhơn |
| Đánh giá Khóa luận   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm   * Một số nhận xét về hình thức cuốn báo cáo:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………   1. Về nội dung nghiên cứu:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………   1. Về chương trình ứng dụng:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………   1. Về thái độ làm việc của sinh viên:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………  Đánh giá chung:  ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………  Điểm từng sinh viên:  Lương Quốc Khang :………../10 | | |

|  |  |
| --- | --- |
|  | Người nhận xét  (Ký tên và ghi rõ họ tên) |

|  |  |
| --- | --- |
| **ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | **TP. HCM, ngày…..tháng…..năm……..** |

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**

**(CỦA CÁN BỘ PHẢN BIỆN)**

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **ỨNG DỤNG HỖ TRỢ THI TOÁN TRÁC NGHIỆM THPT ONLINE** | | |
| Nhóm SV thực hiện: | | Cán bộ phản biện: |
| Lương Quốc Khang | 15520342 |  |
| Đánh giá Khóa luận   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm   * Một số nhận xét về hình thức cuốn báo cáo:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………   1. Về nội dung nghiên cứu:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………   1. Về chương trình ứng dụng:   ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ………………………………………………………………………………………………………   1. Về thái độ làm việc của sinh viên:   ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………  Đánh giá chung:  ……………………………………………………………………………………………………… ……………………………………………………………………………………………………… ………………………………………………………………………………………………………  Điểm từng sinh viên:  Lương Quốc Khang :………../10 | | |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | Người nhận xét  (Ký tên và ghi rõ họ tên) |

|  |  |
| --- | --- |
| **ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | TP. HCM, ngày…..tháng…..năm…….. |

ĐỀ CƯƠNG CHI TIẾT

|  |  |
| --- | --- |
| TÊN ĐỀ TÀI:  Ứng dụng Web hỗ trợ học thi Toán THPT theo hình thức trắc nghiệm  High school Math test in the form of multiple choice web application | |
| Cán bộ hướng dẫn: PGS.TS. Đỗ Văn Nhơn | |
| Thời gian thực hiện:Từ ngày 14/10/2019 đến ngày 6/1/2020 | |
| Sinh viên thực hiện:  Lương Quốc Khang - 15520342 | |
| Nội dung đề tài:   * Mục tiêu : Tìm hiểu về công nghệ và công cụ phù hợp; thiết kế, cài đặt và thử nghiệm một ứng dụng Web dựa trên một ngân hàng câu hỏi trắc nghiệm cho (một phần) kiến thức Toán THPT. Hệ thống cho phép tạo đề tự động theo yêu cầu có liên quan đến các chủ đề hay nội dung kiến thức, cho học sinh làm bài, chấm điểm và khuyến nghị. * Phạm vi: phần mềm nhắm đến đối tượng học sinh lớp 12 học và chuẩn bị cho kỳ thi trung học phổ thông và thi đại học * Phương pháp thực hiện : khảo sát thực tế , thu tập các tài liệu và bộ đề thi. Xây dựng ứng dụng trên nên tảng web nhằm giúp cho việc học tập trở nên dễ dàng và thuận tiện hơn * Kết quả mong đợi : Xây dựng website giúp các bạn học sinh có thể dễ dàng tiếp cận với cấu trúc đề thi trung học phổ thông theo hình thức trắc nghiệm. Qua đó các bạn học sinh có thể dánh giá được năng lực của mình hiện tại và website sẽ đưa ra các đề xuất để cải thiện | |
| Kế hoạch thực hiện:  Lương Quốc Khang :  Tuần 1 ( 14/10/2019 – 20/10/2019 ) : xác định đề tài và viết đề cương chi tiết  Tuần 2 ( 21/10/2019 – 27/10/2019 ) : định hướng đề tài, quyết định các tính năng và đánh giá độ ưu tiên  Tuần 3 ( 28/10/2019 – 3/11/2019 ) : phân tích các tính năng, thu thập tài liệu , đề bài  Tuần 4 ( 4/11/2019 – 10/11/2019 ) : xây dựng wireframe , giao diện và cơ sở dữ liệu  Tuần 5 6 7 8 ( 11/11/2019 – 8/12/2019 ) : Lập trình các tính năng  Tuần 9 10 ( 9/12/2019 – 22/12/2019 ) : Kiểm thử các tính năng và viết báo cáo khóa luận | |
| Xác nhận của CBHD | TP. HCM, ngày 18 tháng 10 năm 2019  Sinh viên |

**LỜI CÁM ƠN**

Em xin gửi lời cảm ơn chân thành tới giảng viên hướng dẫn PGS. TS Đỗ Văn Nhơn đã tận tình hướng dẫn và giúp đỡ cho chúng em trong thời gian qua để hoàn thiện đề tài này.

Em cũng xin cảm ơn các thầy cô ở trường đại học Công nghệ thông tin nói chung và các thầy cô ở khoa công nghệ phần mềm nói riêng đã giúp đỡ chúng em đến ngày hôm nay.

Một lần nữa, em xin chân thành cảm ơn.

MỤC LỤC

[Chương 1 PHÁT BIỂU BÀI TOÁN 5](#_Toc29143237)

[1.1 Mở đầu: 5](#_Toc29143238)

[1.2 Ứng dụng hỗ trợ thi trắc nghiệm toán online: 5](#_Toc29143239)

[1.3 Lợi ích và thách thức: 5](#_Toc29143240)

[1.3.1 Lợi ích: 5](#_Toc29143241)

[1.3.1.1 Đối với học sinh, sinh viên: 5](#_Toc29143242)

[1.3.1.2 Đối với giảng viên: 6](#_Toc29143243)

[1.3.2 Thách thức: 6](#_Toc29143244)

[1.4 Kết chương: 6](#_Toc29143245)

[Chương 2 TỔNG QUAN CÔNG NGHỆ 7](#_Toc29143246)

[2.1 Tổng quan về mô hình MVC 7](#_Toc29143247)

[2.1.1 Giới thiệu về mô hình MVC 7](#_Toc29143248)

[2.1.2 Các thành phần của MVC 8](#_Toc29143249)

[2.1.3 Hoạt động của mô hình MVC 8](#_Toc29143250)

[2.1.4 Ưu điểm và nhược điểm của MVC 9](#_Toc29143251)

[2.2 ASP.NET Core MVC Framework 11](#_Toc29143252)

[2.2.1 Hoàn Cảnh ra đời của ASP.NET Core? 11](#_Toc29143253)

[2.2.2 Giới thiệu chung 11](#_Toc29143254)

[2.2.3 Các thành phần cơ bản của ASP.NET Core 13](#_Toc29143255)

[2.2.3.1 Cấu hình trong ASP.NET Core 13](#_Toc29143256)

[1.1.1.1 Controller và Action Results 14](#_Toc29143257)

[2.2.3.2 Views 18](#_Toc29143258)

[2.2.3.3 Models 21](#_Toc29143259)

[2.2.3.4 URL và Routing 30](#_Toc29143260)

[2.2.3.5 Middleware 36](#_Toc29143261)

[2.2.3.6 Cơ chế Dependency Injection 37](#_Toc29143262)

[2.2.4 Quá trình hoạt động của một ứng dụng ASP.NET Core 41](#_Toc29143263)

[2.2.5 Ưu điểm và nhược điểm 41](#_Toc29143264)

[2.2.6 Làm quen với ứng dụng ASP.NET Core: 43](#_Toc29143265)

[2.2.6.1 Cách khởi tạo dự án ASP.NET Core: 43](#_Toc29143266)

[2.2.6.2 Cấu trúc dự án: 46](#_Toc29143267)

[Chương 3 CHƯƠNG TRÌNH ỨNG DỤNG 49](#_Toc29143268)

[3.1 Mở đầu: 49](#_Toc29143269)

[3.2 Xây dựng chức năng: 50](#_Toc29143270)

[3.2.1 Danh sách các actor 50](#_Toc29143271)

[3.2.2 Phân tích chức năng trang chính: 51](#_Toc29143272)

[3.2.2.1 Đăng nhập: 51](#_Toc29143273)

[3.2.2.2 Đăng xuất 52](#_Toc29143274)

[3.2.2.3 Đăng ký 53](#_Toc29143275)

[3.2.2.4 Xem Danh sách đề thi 54](#_Toc29143276)

[3.2.2.5 Xem chi tiết đề thi 55](#_Toc29143277)

[3.2.2.6 Làm bài thi toán online 57](#_Toc29143278)

[3.2.2.7 Tìm kiếm 58](#_Toc29143279)

[3.2.2.8 Xem điểm và nội dung bài làm của các đề thi 59](#_Toc29143280)

[3.2.3 Trang admin 61](#_Toc29143281)

[3.2.3.1 Xóa người dung 61](#_Toc29143282)

[3.2.3.2 Cập nhật thông tin người dung 63](#_Toc29143283)

[3.2.3.3 Thêm đề thi 65](#_Toc29143284)

[3.2.3.4 Xóa đề thi 67](#_Toc29143285)

[3.2.3.5 Cập nhật thông tin đề thi 68](#_Toc29143286)

[3.2.3.6 Thêm câu hỏi: 73](#_Toc29143287)

[3.3 Các công cụ và thư viện sử dụng: 78](#_Toc29143288)

[3.4 Kết chương: 78](#_Toc29143289)

[Chương 4 Thử nghiệm và đánh giá 79](#_Toc29143290)

[4.1 Mở đầu: 79](#_Toc29143291)

[4.2 Dữ liệu thử nghiệm: 79](#_Toc29143292)

[4.3 Kết quả: 81](#_Toc29143293)

[4.4 Kết chương: 82](#_Toc29143294)

[Chương 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 83](#_Toc29143295)

**DANH MỤC HÌNH VẼ**

[Hình 2.1: Mô hình MVC 8](#_Toc29143296)

[Hình 2.2: Quy trình hoạt động của mô hình MVC 9](#_Toc29143297)

[Hình 2.3: Vị trí của .NET core trong mô hình phát triển .NET 12](#_Toc29143298)

[Hình 2.4: Cách tạo controller 15](#_Toc29143299)

[Hình 2.5: Cách tạo controller 16](#_Toc29143300)

[Hình 2.6: Minh họa Razor view 19](#_Toc29143301)

[Hình 2.7: Hướng dẫn tạo View 20](#_Toc29143302)

[Hình 2.8: Hướng dẫn tạo View 21](#_Toc29143303)

[Hình 2.9: Hướng dẫn tạo View cách 2 21](#_Toc29143304)

[Hình 2.10: ví dụ về domain model 22](#_Toc29143305)

[Hình 2.11: Cơ chế Model Binding 24](#_Toc29143306)

[Hình 2.12: Minh họa cơ chế Model Binding 25](#_Toc29143307)

[Hình 2.13: Mô hình hoạt động của model validation 29](#_Toc29143308)

[Hình 2.14: Thông báo lỗi model valitaion 30](#_Toc29143309)

[Hình 2.15: Routing 31](#_Toc29143310)

[Hình 2.16: Cách routing hoạt động 32](#_Toc29143311)

[Hình 2.17: URL pattern 34](#_Toc29143312)

[Hình 2.18: Url Matching 35](#_Toc29143313)

[Hình 2.19: Cách middleware hoạt động 36](#_Toc29143314)

[Hình 2.20: Method không sử dụng DI 38](#_Toc29143315)

[Hình 2.21: Moethod sử dụng DI 39](#_Toc29143316)

[Hình 2.22: Cách hoạt động của dependency injection 40](#_Toc29143317)

[Hình 2.23: Vòng đời của ứng dụng asp.net core 41](#_Toc29143318)

[Hình 2.24: So sánh số lượng Request trong 1 giây 42](#_Toc29143319)

[Hình 2.25: Mở hộp thoại tạo project ASP.NET Core ( 1 ) 43](#_Toc29143320)

[Hình 2.26: Đặt tên project 44](#_Toc29143321)

[Hình 2.27: Mô hình MVC trong project 45](#_Toc29143322)

[Hình 2.28: Chạy project 45](#_Toc29143323)

[Hình 2.29: Cấu trúc của một dự án ASP.NET Core 46](file:///C:\Users\PPO\Downloads\BÁO%20CÁO%20.docx#_Toc29143324)

[Hình 2.30: Cấu trúc launchSetting 47](#_Toc29143325)

[Hình 3.1: sequence diagram đăng nhập 52](#_Toc29143326)

[Hình 3.2: sequence diagram đăng ký 54](#_Toc29143327)

[Hình 3.3: Sequence diagram xem danh sách đề thi 55](#_Toc29143328)

[Hình 3.4: Sequence diagram xem chi tiết đề thi 56](#_Toc29143329)

[Hình 3.5: Sequence diagram làm bài thi toán 58](#_Toc29143330)

[Hình 3.6: Sequence diagram tìm kiếm 59](#_Toc29143331)

[Hình 3.7: Xem điểm bài thi 61](#_Toc29143332)

[Hình 3.8: Sequence diagram xóa người dung 63](#_Toc29143333)

[Hình 3.9: Cập nhật thông tin người dung 65](#_Toc29143334)

[Hình 3.10: Sequence diagram thêm đề thi 67](#_Toc29143335)

[Hình 3.11: Sequence diagram thêm câu hỏi vào đề thi 71](#_Toc29143336)

[Hình 3.12: Sequence diagram xóa câu hỏi khỏi để thi 73](#_Toc29143337)

[Hình 3.13: Sequence diagram thêm câu hỏi 75](#_Toc29143338)

[Hình 3.14: Sequence diagram xóa câu hỏi 76](#_Toc29143339)

[Hình 3.15: Cập nhật thông tin câu hỏi 78](#_Toc29143340)

[Hình 4.1: Danh sách đề thi 79](#_Toc29143341)

[Hình 4.2: Đề thi mẫu 80](#_Toc29143342)

[Hình 4.3: Trang làm bài kiếm tra 81](#_Toc29143343)

[Hình 4.4: Kết quả 81](#_Toc29143344)

DANH MỤC BẢNG

[Bảng 2.1: Kiểu dữ liệu trả về của IActionResult 18](#_Toc29143345)

[Bảng 2.2: Default model binder 26](#_Toc29143346)

[Bảng 2.3 Các loại middleware 37](#_Toc29143347)

[Bảng 2.4: chức năng của các file sau khi tạo project 48](#_Toc29143348)

[Bảng 3.1: Danh sách các tính năng 50](#_Toc29143349)

[Bảng 3.2: Danh sách các actor 51](#_Toc29143350)

[Bảng 3.3: Dăng nhập 51](#_Toc29143351)

[Bảng 3.4: Đăng xuất 53](#_Toc29143352)

[Bảng 3.5: Đăng ký 54](#_Toc29143353)

[Bảng 3.6: Xem danh sách đề thi 55](#_Toc29143354)

[Bảng 3.7: Xem chi tiết đề thi 56](#_Toc29143355)

[Bảng 3.8: Làm bài thi toán online 57](#_Toc29143356)

[Bảng 3.9: TÌm kiếm 59](#_Toc29143357)

[Bảng 3.10: Xem điểm và nội dung bài làm của các đề thi đã thi 60](#_Toc29143358)

[Bảng 3.11 Xóa người dung 62](#_Toc29143359)

[Bảng 3.12: Cập nhật thông tin người dung 64](#_Toc29143360)

[Bảng 3.13: Thêm đề thi 67](#_Toc29143361)

[Bảng 3.14: Xóa đề thi 68](#_Toc29143362)

[Bảng 3.15: Cập nhật thông tin chung 69](#_Toc29143363)

[Bảng 3.16: Thêm câu hỏi vào đề thi 70](#_Toc29143364)

[Bảng 3.17: Xóa câu hỏi của đề thi 72](#_Toc29143365)

[Bảng 3.18: Thêm câu hỏi 74](#_Toc29143366)

[Bảng 3.19; Xóa câu hỏi 76](#_Toc29143367)

[Bảng 3.20: Cập nhật nội dung và câu trả lời 77](#_Toc29143368)

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| Thuậ ngữ/ Từ viết tắt | Nghĩa đầy đủ |
| Class | Lớp |
| Request | Yêu cầu |
| Action method | Phương thức hoạt động |
| Client | Phía người dung |
| Server | Máy chủ |
| Code | Mã lệnh |
| Namspace | Không gian tên |
| Browser | Trình duyệt |
| Framework | Nền tảng |
| Filter | Bộ lọc |
| Validation | Xác nhận |
| MVC | Model-View-Controller |
| CSDL | Cơ sở dữ liệu |
| HTML | Hypertext Markup Language |
| HTTP | HyperText Tranfer Protocol |
| XSS | Cross Site Scripting |
| URL | Uniform Resource Locator |

TÓM TẮT KHÓA LUẬN

Xây dụng Website trên nên tảng ASP.NET Core tích hợp với angular 5 để mang lại trải nghiệm tốt nhất cho người sử dụng. Với mô hình MVC giúp tách biệt các thành phần của ứng dụng. Lập trình viên có thể dễ dàng xây dụng website với cấu trúc chặt chẽ, rành mạch. Ngoại ra nó cũng giúp việc sửa lỗi, thay thế và thử nghiệm các chức năng của website trở nên dễ dàng hơn

Với những ưu điểm trên, ASP.NET Core sẽ gần như là nền tảng chính trong việc xây dựng và phát triền website. Vì vậy, trong khóa luận tốt nghiệp này em sẽ tìm hiểu về công nghệ ASP.NET Core của Microsoft và xây dựng một ứng dụng dựa trên công nghệ này

MỞ ĐẦU

1. **Dẫn nhập:**

Ngày nay kỳ thi trung học phổ thông quốc gia đã trở nên vô cùng quan trọng và cũng là một bước ngoặc đối với cuộc đời mỗi học sinh. Môn toán là một phần rất quan trọng trong các khối tự nhiên và hiện nay đã chuyển sang hình thức thi trắc nghiệm.

Để giải quyết vấn đề trên website ra đời nhằm hổ trợ học sinh trong việc luyện tập kỹ năng , năng cao kiến thức và thống kê kết quả học tập cho học viên với mục đích giúp học viên nhận ra điểm chưa tốt và cải thiện.

1. **Mục tiêu:**

* Tìm hiểu về một công nghệ mới
* Xây dựng website hỗ trợ học sinh luyện tập thi môn toán với hình thức trắc nghiệm
* Cung cấp tài liệu tiếng Việt về công nghệ này

1. **Phạm vi đề tài:**

* Nắm được các kiến thức cơ bản về ASP.NET Core: cách thức hoạt động và giao tiếp đến các công nghệ khác

1. **Phương pháp nghiên cứu:**

* Nghiên cứu lý thuyết thông qua ebook của Microsoft và các diễn đàn
* Từng bước tìm hiểu các ví dụ cơ bản, sau đó áp dụng vào một ứng dụng nhỏ. Từ đó tổng hợp kiến thức và xây dựng ứng dụng hoàn chỉnh
* Xây dựng các demo trong khi nghiên cứu
* Đánh giá kết quả

1. **Bố cục báo cáo:**

Bài báo cáo này gồm 5 chương với nội dung như sau :

* Chương 1 – Phát biểu bài toán : nêu các vấn đề hiện tại và tìm hiểu phương pháp giải quyết vấn đề đã đặt
* Chương 2 - Tổng quan về công nghệ sử dụng: giới thiệu về mô hình MVC cũng như đưa ra cái nhìn tổng quan về công nghệ ASP.NET Core, angular và cách tích hợp 2 công nghệ này lại với nhau
* Chương 3 – Chương trình ứng dụng: xây dựng cá thành phần và tính năng của hệ thống
* Chương 4 – Thử nghiệm và đánh giá: thực thi hệ thống, kiểm tra và đánh giá tính đúng đắn, tính tiện dụng và hiệu suất của hệ thống
* Chương 5 – Kết luận và hướng phát tiển: tổng kết các kết quả đạt được trong quá trình thực hiện và đề xuất các khía cạnh có thể mở rộng đề tài

# PHÁT BIỂU BÀI TOÁN

## Mở đầu:

Chương này sẽ tìm hiểu tổng quan về vấn đề được đặt ra, đề ra một số yêu cầu ban đầu của ứng dụng, phân tích các lợi ích của nó cũng như những thách thức trong việc xây dựng

## Ứng dụng hỗ trợ thi trắc nghiệm toán online:

Ứng dụng hỗ trợ thi trắc nghiệm toán online là ứng dụng cho phép mô phổng bài thi toán trung học phổ thông với hình thức trắc nghiệm dưới dạng bài làm trên máy tính. Học sinh có thể tham gia thi trực tuyến. Hệ thống sẽ chấm điểm dựa trên các câu trả lời của học sinh từ đó đưa ra biểu đồ thống kê số lượng câu sai , đúng theo từng dạng câu hỏi và đưa ra các bài tập , lời khuyên, đề thi thử nhằm giúp học sinh cải thiện những phần chưa tốt. Không chỉ dừng lại ở đó, công cụ này còn giúp các quản trị viên học giảng viên có thể tạo ra các đề thi cho học sinh nhằm giúp đa dạng hóa bài tập và có thể tạo ra các đề thi tập trung vào việc cải thiện từng dạng bài.

Hiện nay cũng có nhiều công cụ hỗ trợ việc thi toán trắc nghiệm online nhưng còn gặp nhiều hạn chế chức năng hay đối tượng. Đây là giải pháp giúp cho học sinh cũng như giảng viên học tập một cách hiệu quả trong thời đại 4.0

Nội dung các chức năng chính sẽ được mô tả chi tiết trong phần nội dung khóa luận

## Lợi ích và thách thức:

### Lợi ích:

#### Đối với học sinh, sinh viên:

* Dễ dàng tiếp cận kiến thức hơn so với cách truyền thống
* Có thể xem được sự tiến bộ của mình qua từng bài kiểm tra
* Giúp học sinh định hướng cải thiện những dạng bài chưa tốt
* Học sinh có thể dễ dàng làm bài kiểm tra mọi lúc mọi nơi

#### Đối với giảng viên:

* Việc giảng dạy dễ dàng hơn nhờ bộ đề được tạo từ công cụ này
* Sử dụng trên web nên chỉ cần mạng internet là có thể kết nối, không cần giấy bút như kiểu truyền thống

### Thách thức:

* Cần nguồn dữ liệu lớn các câu hỏi toán học và lời giải chính xác
* Phụ thuộc vào dữ liệu

## Kết chương:

Qua chương này ta thu thập được những kiến thức nền tảng về ứng dụng, các ưu khuyết, áp dụng và đánh giá ứng dụng hỗ trợ thi trắc nghiệm toán Thpt theo hình thức trắc nghiệm. Những kết quả này là bước đầu để đi sâu vào nghiên cứu các công nghệ, thuật toán áp dụng nó vào đề tài.

# TỔNG QUAN CÔNG NGHỆ

## Tổng quan về mô hình MVC

### Giới thiệu về mô hình MVC

Khi các phần mềm ứng dụng ngày càng phát triển lớn mạnh và đa dạng thì vấn đề thiết kế kiến trúc cho phần mềm cũng trở nên phức tạp hơn. Những ứng dụng web giờ đây cũng đã phát triển đến một tầm cao mới, chúng không còn là những ứng dụng web thông thường với chức năng hiển thị thông tin mà là những hệ thống thông tin hỗ trợ những trang web tương tác với người dùng. Nhiều giải pháp thiết kế kiến trúc hệ thống được đưa ra với mục đích làm cho hệ thống linh hoạt, mềm dẻo, dễ dễ phát triển và tồn tại lâu dài. Trong đó giải pháp thiết kế theo mô hình kiến trúc MVC được Trygve ReensKaug đề xuất vào những năm 70 của thế kỷ 20 to ra rất hiệu quả.

Mô hình MVC ( Model – View – Controller ) là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó giúp cho các nhà phát triển tách ứng dụng của họ ra thành 3 phần khác nhau Model, View và Controller. Mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.

### Các thành phần của MVC



Hình 2.1: Mô hình MVC

**Model**

Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tưỡng mô tả dữ liệu như các class, hàm xử lý,…

**View**

Đảm nhận việc hiển thị thông tin, tương tác với người dung.

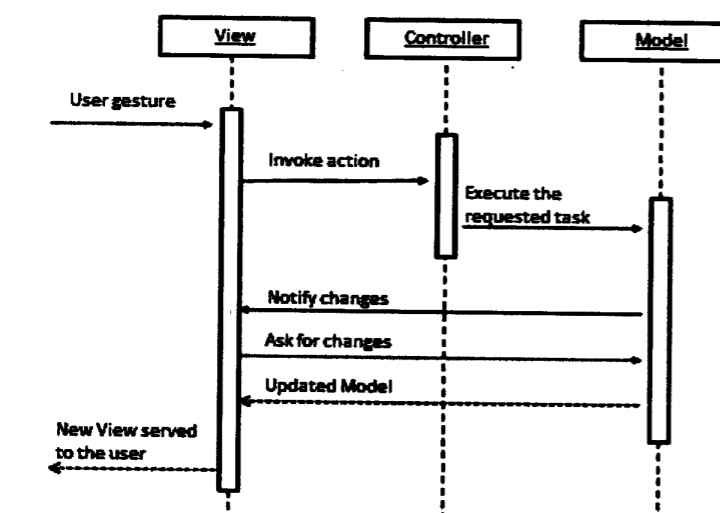
**Controller**

Giữ nhiệm vụ điều hướng các yêu cầu từ người dung và gọi đúng những phương thức xử lý chúng.

### Hoạt động của mô hình MVC

Đây là một cách đơn giản để mô tả luồng sự kiện được xử lý trong MVC

* + User tương tác với View, bằng cách click button, user gửi yêu cầu đi.
  + Controller nhận và điều hướng chúng đến đúng phương thức xử lý ở Model.
  + Model nhận thông tin và thực thi yêu cầu.
  + Khi model hoàn tất việc xử lý, View sẽ nhận dữ liệu từ Model và Controller sẽ phản hồi lại người dùng



Hình 2.2: Quy trình hoạt động của mô hình MVC

### Ưu điểm và nhược điểm của MVC

**Ưu điểm:**

* Phát triển nhanh chóng, duy trì dễ dàng :
  + MVC chia mọi hệ thống làm 3 phần hoạt động tương đối độc lập với nhau. Vì vậy, có thể chi làm 3 team lập trình viên phát triển cùng lúc 3 thành phần này. Điều này cực kỳ cần thiết cho những dự án, phần mềm quy mô lớn, phức tạp.
  + Mô hình MVC cũng đồng thời phù hợp với lý thuyết lập trình separation of concerns (SoC), tức chia nhỏ hệ thống thành nhiều phần để dễ dàng giải quyết các vấn đề. Khi có lỗi phát sinh, các lập trình viên có thể nhanh chóng định vị, cô lập và chỉnh sửa được phần gặp lỗi.
* Thay đổi chỉnh sửa không làm ảnh hưởng tới toàn bộ hệ thống:
  + Không chỉ nhanh chóng phát hiện bộ phận bị lỗi, MVC còn cho phép chỉnh sửa các thành phần mà không làm ảnh hưởng tới toàn bộ hệ thống. Có thể tách Model ra để bảo trì, chỉnh sửa các API bên trong mà không thay đổi cách hiển thị trên View.
  + Đặc điểm này cũng giúp ích rất nhiều trong quá trình kiểm thử (test), cho phép đơn vị kiểm thử hoạt động tự do, độc lập, không cần tạm ngưng toàn bộ hệ thống chỉ để kiểm tra một chức năng nào đó.
* Hỗ trợ đa dạng các framework, dễ dàng mở rộng, cải tiến
* Thân thiện cới SEO : Các hoạt động của MVC hỗ trợ chức năng URL Routing, giúp xây dựng những URL than thiện với SEO – một ưu điểm tuyệt vời đối với lập trình, thiết kế website.

**Nhược điểm:**

* Việc chia hệ thống làm 3 phần và tương tác giữa mỗi phần với nhau khiến MVC quá cồng kềnh cho những phần mềm nhỏ gọn. Trong đó, lớp Controller cần duy trì một số lượng code lớn để vừa ghi nhận user request vừa quyết định phương thức xử lý chúng, cũng như nhiều tác vụ khác
* Sự hỗ trợ cho quá trình kiểm thử của MVC cũng không quá tốt. Lý do là bởi lớp View phụ thuộc vào cả Model lẫn Controller.
* Mô hình MVC thiết kế mất cân bằng khiến Model dễ gặp lỗi hơn, phát triển khó hơn và sửa chữa, bảo trì rắc rối hơn. Những lập trình viên lo phần Model sẽ phải làm việc vất vả, cũng như cần chuyên môn và sự tỉ mỉ lớn hơn.

## ASP.NET Core MVC Framework

### ****Hoàn Cảnh ra đời của ASP.NET Core?****

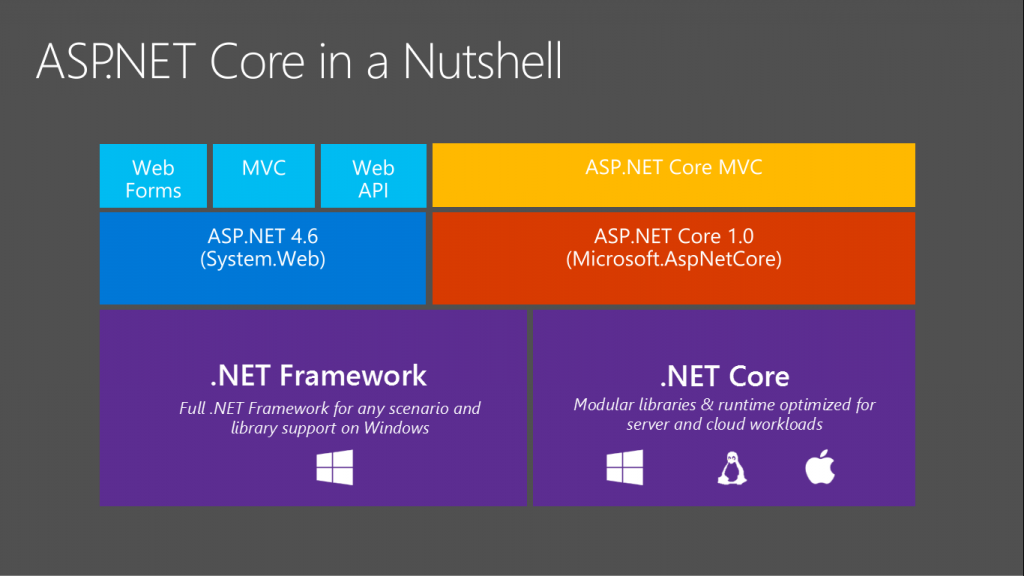
Bản phát hành đầu tiên của ASP.NET đã xuất hiện cách đây 15 năm trước, nó là một phần của .NET Framework. Từ đó, hàng triệu lập trình viên đã sử dụng nó để xây dựng những ứng dụng web tuyệt vời, và trên những năm đó Microsoft đã phát triển thêm nhiều tính năng mới.

ASP.NET Core có một số thay đổi kiến trúc lớn, đó là kết quả của việc học hỏi rất nhiều từ các framework module hóa khác. ASP.NET Core không còn dựa trên System.Web.dll nữa. Nó được dựa trên một tập hợp các gói, các module hay cũng được gọi là các Nuget packages. Điều này cho phép bạn tối ưu ứng dụng của bạn để chỉ bao gồm những packages nào cần thiết. Lợi ích của nó là giúp cho ứng dụng nhỏ hơn, bảo mật chặt chẽ hơn, giảm sự phức tạp, tối ưu hiệu suất hoạt động và giảm chi phí, thời gian cho việc phát triển.

### Giới thiệu chung

ASP.NET Core là một nên tảng mã nguồn mở, cross-platform framework cho việc xây dựng và kết nối các ứng dụng web hiện đại chẳng hạn như ứng dụng web, Internet of thing, Mobile backend – web API

ASP.NET Core cung cấp một kiến trúc để tối ưu hóa việc xây dựng các ứng dụng đám mây ( Microsoft azure ) hoặc các ứng dụng độc lập. CÓ thể phát triển ASP.NET Core đa nền tảng ( Window, Linux, Mac )



Hình 2.3: Vị trí của .NET core trong mô hình phát triển .NET

ASP.NET Core còn được xem là sự kết hợp giữa ASP.NET [MVC](https://www.dammio.com/glossary/mvc) và ASP.NET Web API tạo thành một mô hình lập trình đơn. Mặc dù được xây dựng mới, ASP.NET Core vẫn có tính tương thích cao với ASP.NET [MVC](https://www.dammio.com/glossary/mvc). Hơn nữa, các ứng dụng ASP.NET Core hỗ trợ kiểu phiên bản “side by side”, tức là cùng chạy trên một máy tính với việc lựa chọn nhiều phiên bản ASP.NET Core khác nhau. Điều này là không thể với các phiên bản ASP.NET trước kia.

Với ASP.NET Core, bạn có thể:

* Xây dựng các ứng dụng web, các dịch vụ, ứng dụng IoT và các phần backend mobile.
* Sử dụng các công cụ phát triển ưa thích trên Windows, macOS và Linux.
* Triển khai trên đám mây hoặc tại chỗ.
* Chạy trên .NET Core hoặc .NET Framework.

### Các thành phần cơ bản của ASP.NET Core

#### Cấu hình trong ASP.NET Core

Cấu hình là các tham số hoặc các cài đặt cụ thể cho ứng dụng. Các cài đặt này được lưu trữ riêng biệt trong code và trong các file độc lập. Nó giúp các developer và quản trị kiểm soát và dễ dàng thay đổi cách mà ứng dụng chạy.

Ví dụ: Connection Strings cần để kết nối đến cơ sở dữ liệu được lưu trong một file cấu hình. Bằng cách thay đổi chuỗi connection mà bạn có thể thay đổi tên cơ sở dữ liệu, vị trí… mà không cần thay đổi mã nguồn.

Cấu hình là cách lưu các thông tin cần thiết, cung cấp cho ứng dụng khi nó chạy. Đối với ứng dụng ASP.NET các file cấu hình có thể lưu ở các định dạng XML, JSON, INI tùy cách lựa chọn. Hiện nay, mặc định ứng dụng ASP.NET core sẽ nạp file cấu hình JSON lưu ở appsettings .json nếu nó tồn tại.

Cấu hình trong ASP.NET Core:

Cấu hình trong ứng dụng ASP.NET Core được lưu dưới dạng cặp tên-giá trị. Bạn có thể lưu chúng trong file JSON, XML hay INI.

{

"option1": "value1",

"option2": 2

}

Đây cũng là điểm mới và mạnh hơn so với sử dụng Web.Config trước đây khi mà các cấu hình được nhóm lại từng nhóm và hỗ trợ nhiều cấp giúp quản lý cấu hình dễ dàng hơn. Ứng dụng ASP.NET Core đọc các file cấu hình từ lúc khởi động. Nó có thể được cấu hình để đọc trong lúc chạy nếu nó thay đổi.

ASP.NET Core hỗ trợ đọc cấu hình từ các nguồn khác nhau và các định dạng khác nhau. Một vài nguồn được sử dụng phổ biến như:

* Định dạng file (JSON, INI hoặc XML)
* Command line Arguments (tham số dòng lệnh)
* Environment variables (biến môi trường)
* Custom Provider (cái này là tự tạo ra provider riêng theo ý muốn)

#### Controller và Action Results

**⁂ Controller:**

**Controller** là thành phần đầu tiên nhận request từ người dùng. Khi người dùng truy cập URL qua trình duyệt, ASP.NET Core routing sẽ map request đó vào controller cụ thể.

Ví dụ: Request URL như sau: **http://localhost/Customer/List**

Trường hợp này, Controller có tên là **CustomerController**được gọi. Sau đó nó sẽ gọi đến Action metod tên **List**rồi tạo ra response trả về cho user.

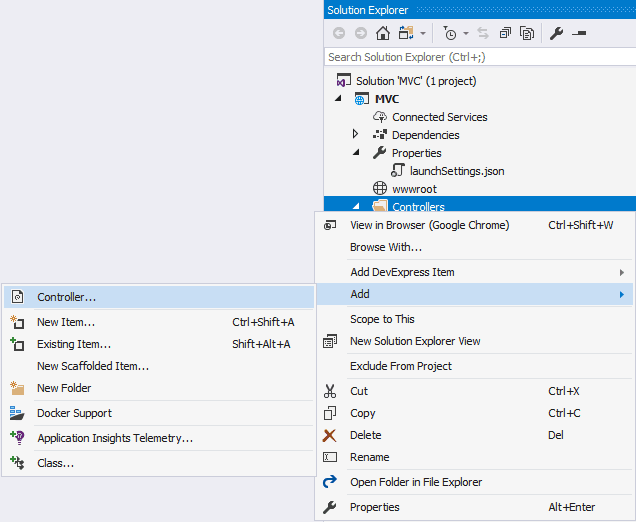
**Trách nhiệm của Controller:**

Controller có 3 trách nhiệm chính:

* **Nhận request**: Controller có trách nhiệm nhận request từ user.
* **Dựng model**: Controller Action method thực thi logic của ứng dụng và xây dựng nên model.
* **Gửi trả response**: Cuối cùng nó sẽ trả về kết quả trong HTML, File, JSON, XML hoặc bất cứ định dạng nào về cho user.

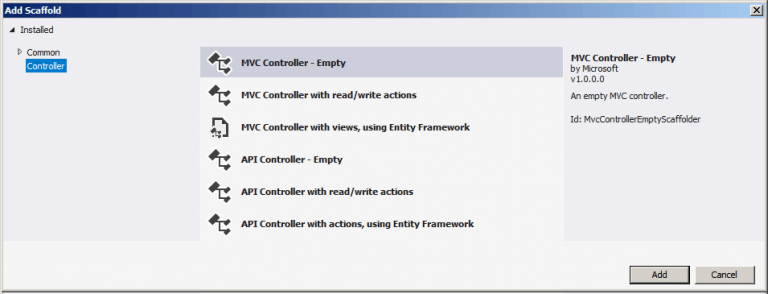
**Thêm mới controller**

Controller không khác gì so với .NET Class thông thường. Chỉ có điều nó nằm trong thư mục **Controllers**là thư mục nằm trong thư mục gốc của ứng dụng. Bạn có thể tạo mới controller bằng cách chọn thư mục **Controllers**và chuột phải chọn **Add-> Controller.**

****

Hình 2.4: Cách tạo controller

Cả MVC và API controller đều kế thừa từ cùng class Controller nên nó không khác nhau nhiều. Ngoại trừ API controller nhằm mục đích trả về dữ liệu được định dạng cho client.



Hình 2.5: Cách tạo controller

Các controller trong ASP.NET Core kế thừa từ class **Controller**và nó lại kế thừa từ class **ControllerBaseClass**. 2 base class này cung cấp nhiều các phương thức hữu ích.

Controller class phải thỏa mãn ít nhất một trong các điều kiện sau:

* Tên class phải có đuôi "Controller"
* Class kế thừa từ một class cung phải có tên đuôi là "Controller"
* Class được đặt thuộc tính [Controller]

**⁂ Action Method:**

Bất cứ public method nào được chìa ra ngoài bởi Controller đều được gọi là Action method. Action method get dduocj gọi khi user gõ một URL trên trình duyệt.

Ví dụ: **http://localhost/Customer/List** sẽ gọi action method tên **List**trong **CustomerController**.

Action method sẽ gọi tầng service layer để phản hồi lại request. Service layer sẽ tương tác với database sử dụng tầng data access layer và map kết quả về model sau đó gửi lại cho Action method.

Action method sẽ gọi View với model để trả về kết quả cho người dùng.

Khi tạo một Action method bạn cần nhớ những điều sau:

* Action method phải là một public method
* Action method không thể là static method hoặc một extension method.
* Constructor, getter, setter không được sử dụng.
* Các phương thức được kế thừa không được sử dụng như là một action method.
* Action method không được chứa từ khóa ref hoặc out trên tham số.
* Action method không được chứa thuộc tính [NonAction]
* Ation method không thể được nạp chồng (overloaded)

**IActionResult và ActionResult**

IActionResult là một interface nó định nghĩa một khuôn mẫu cho toàn bộ các Action Result của một action method. ActionResult là một abstract base class triển khai interface IActionResult. Action result như ViewResult, PartialViewResult hay JsonResult...đều kế thừa từ ActionResult base class.

**Các kiểu dữ liệu trả về của ActionResult**

|  |  |
| --- | --- |
| Tên | Hành động |
| ContentResult | Trả về một chuỗi string |
| FileContentResult | Trả về nội dung file |
| FilePathResult | Trả về nội dung file |
| FileStreamResult | Trả về nội dung file |
| EmptyResult | Không trả về gì cả |
| JavaScriptResult | Trả về đoạn thực thi javascript |
| JsonResult | Trả về dữ liệu dạng json |
| RedirectToResult | Điều hướng đến một URL cụ thể |
| HttpUnauthorizedResult | Trả về mã lội 403 chưa xác thực |
| ViewResult | Nhận phản hồi cho View |

Bảng 2.1: Kiểu dữ liệu trả về của IActionResult

#### Views

Trong mô hình web động thì View là phần hiển thị trang web. Về mặt bản chất, View là một file HTML có chứa các đoạn code ( @ đối với razor ) qua server xử lý để trả về cho trình duyệt ( browser ) để hiển thị

Controller trong ASP.NET Core sẽ nhận request sau đó thực thi với logic tương ứng với dữ liệu đầu vào từ request. Sau đó nó trả về Model cho View. View có trách nhiệm tạo ra giao diện cho người dung từ model

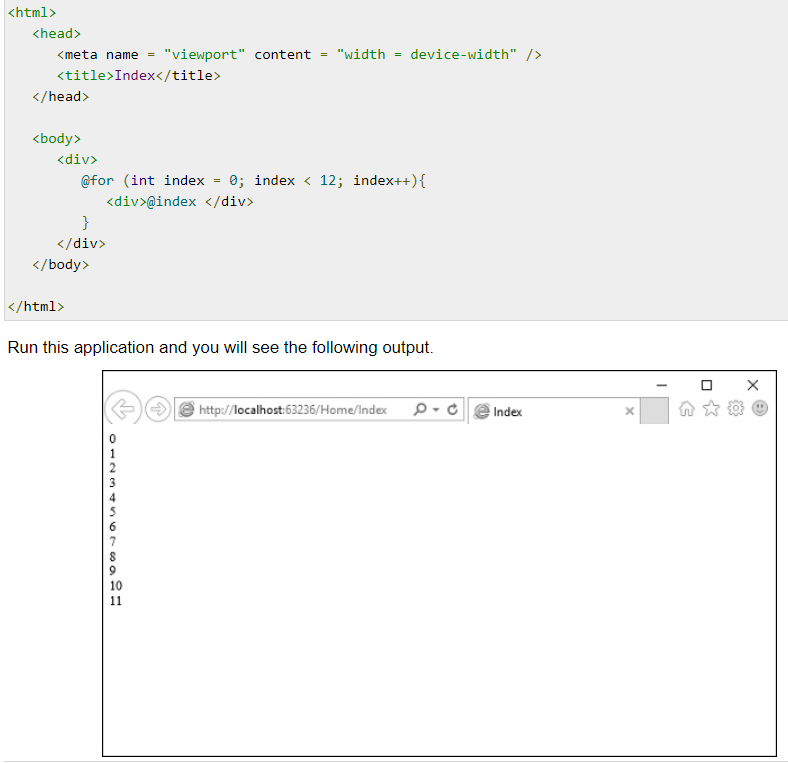
**Giới thiệu về Razor:**

Razor là một ViewEngine tiên tiến của Microsoft, xuất hiện từ phiên bản ASP.NET MVC ( 1/2011). Hầu như các chức năng mà MVC yêu cầu đều có sẵn trong Razor. Razor được biết đến như là ViewEngine mới để thay thế ViewEngine cũ ( ASPX hay Web Forms ).

Razor Pages (mới trong ASP.NET Core 2.0) là một mô hình lập trình dựa trên trang giúp xây dựng giao diện người dùng web dễ dàng hơn và hiệu quả hơn.  
 Razor markup cung cấp một cú pháp hiệu quả cho các trang Razor và các khung nhìn MVC.

Controller trong MVC gọi View bằng cách gán dữ liệu để tạo giao diện. View phải có khả năng xử lý dữ liệu và tạo response. Điều này được xử lý bằng cách dùng Razor, nó cho chúng ta sử dụng C# code trong file HTML. Razor View Engine xử lý các lệnh này và tạo ra HTML.

Các file chứa Razor có đuôi .cshtml. Cú pháp Razor thường ngắn hơn và đơn giản hơn cũng dễ học như C# hoặc VB. Visual Studio IntelliSense cũng hỗ trợ cú pháp Razor.



Hình 2.6: Minh họa Razor view

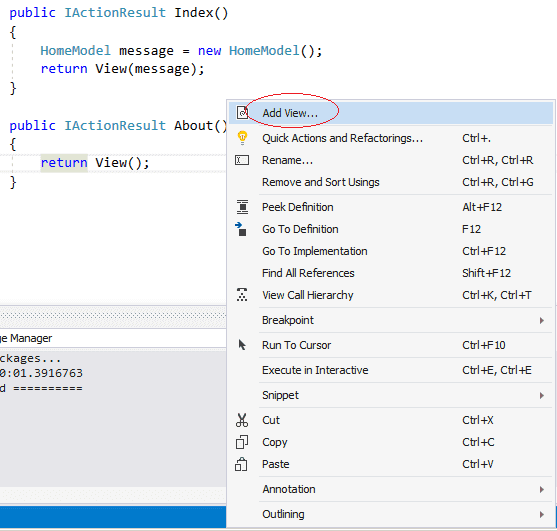
**Ưu điểm của Razor :**

* Gọn nhẹ, xúc tích, dễ dàng: Razor giảm thiểu số ký tự, tỗ hợp phím cần thiết cho một tập tin, và cho phép luồng công việc được thực hiện dễ dàng. Razor

**Cách tạo View trong ASP.NET Core:**

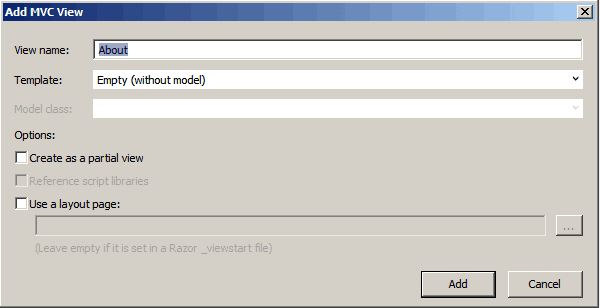
Các method trong Controller trả về một Action **Result**(hoặc một class kế thừa từ ActionResult). Nên tham khảo: [Action Result trong ASP.NET Core](https://tedu.com.vn/lap-trinh-aspnet-core/action-result-trong-aspnet-core-237.html)

ActionResult trả về HTML là một **ViewResult**. Phương thức View của Controller base class trả về một **ViewResult**. Chuột phải vào bất cứ đâu trong phương thức và chọn**Add View**

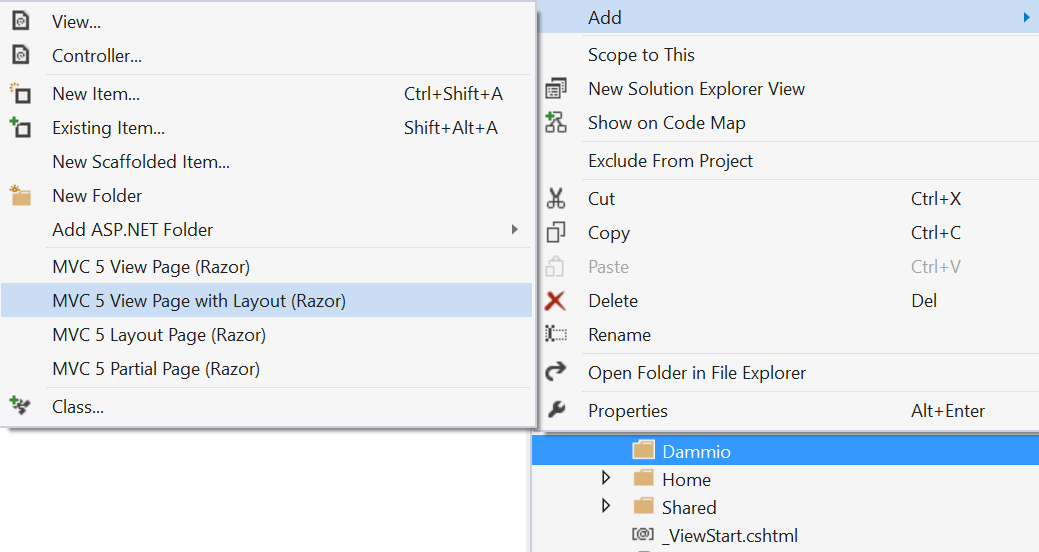


Hình 2.7: Hướng dẫn tạo View

Sau đó sẽ thấy cửa sổ Add MVC View như sau:



Hình 2.8: Hướng dẫn tạo View

****

Hình 2.9: Hướng dẫn tạo View cách 2

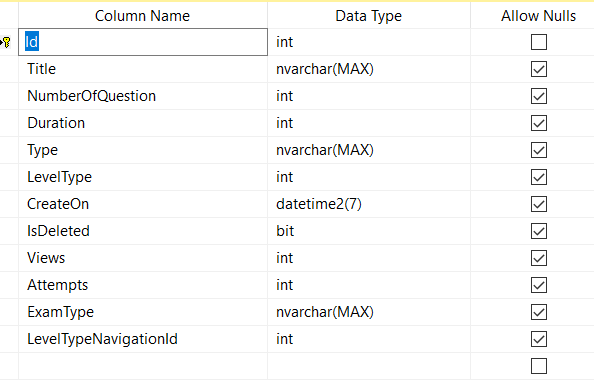
#### Models

Trong mô hình MVC, model chịu trách nhiệm cho phần lõi của ứng dụng và các tác vụ liên quan đến dữ liệu. Cụ thể là kết nối đến cơ sở dữ liệu và thực hiện các thao táo trên dữ liệu đó. Model còn có vai trò mô tả các thực thể trong cơ dữ liệu, kết hợp nhiều loại dữ liệu với nhau theo yêu cầu của View. Ngoài ra còn có thể gắn một model vào view để hiển thị và xác nhận dữ liệu. Có 3 loại mục đích chính:

* Domain Model
* View Model
* Edit Model

**Domain Model là gì ?**

Một Domain Model thể hiện một đối tượng trong database. Domain model thường có một mối quan hệ 1-1 với một bảng  trong cơ sở dữ liệu. Domain Model liên quan đến tầng truy cập dữ liệu (DAL) trong ứng dụng. Nó nhận từ cơ sở dữ liệu hoặc một nơi nào đó lưu dữ liệu bởi tầng truy cập dữ liệu. (DAL). Domain Model cũng được hiểu như entity model hay data model.

****

Hình 2.10: ví dụ về domain model

**View Model là gì ?**

ViewModel được tham chiếu đến các đối tượng chứa dữ liệu cần cho việc hiển thị cho người dùng. ViewModel liên quan đến tầng hiển thị của ứng dụng. Nó được định nghĩa dựa trên cách thức dữ liệu được hiển thị cho người dùng hơn là cách chúng được lưu trữ ra sao.

ViewModel rất hữu dụng khi bạn có một UI phức tạp, khi mà dữ liệu cần lấy ra từ vài Domain Model. Vì View Model được độc lập với Domain Model, nên việc này rất mềm dẻo và linh hoạt cho việc sử dụng nó. ViewModel giúp ứng dụng bảo mật hơn vì bạn không phải chìa các thuộc tính nhạy cảm và bí mật từ Domain Model ra như UserRole,  IsAdmin

**Edit Model là gì ?**

Edit Model hoặc Input Model đại điện dữ liệu cần để người dùng thay đổi hoặc thêm mới. Yêu cầu UI của Product cần chỉnh sửa khác với yêu cầu xem

##### **Model Binding:**

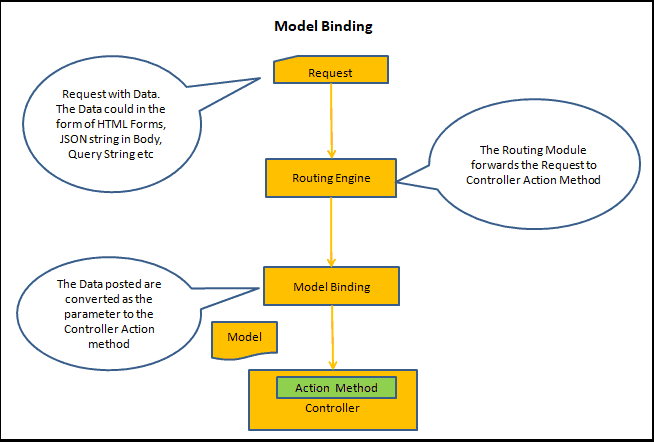
Model binding là quá trình khởi tạo đối tượng .NET sử dụng dữ liệu được gửi bằng trình duyệt ( browser ) trong một yêu cầu HTTP. MVC ánh xạ đến chi tiết của yêu cầu theo cách mà nó có thể truyền dữ liệu thích hợp hoặc tham số đối tượng tới action method, sử dụng model binder mặc định ( DefaultModelBinder ).

Cơ chế ASP.NET Core model binding cho phép chúng ta dễ dàng bind các giá trị này vào các tham số của action method. Các tham số này có thể là kiểu nguyên thủy hoặc kiểu đối tượng phức tạp

ASP.NET Core cho phép chúng ta bind dữ liệu từ nhiều nguồn khác nhau như HTML Form sử dụng **[FromForm]**, từ giá trị route **[FromRoute]**, từ query string **[FromQuery]**, từ  body của request**[FromBody]** và từ Header của request **[FromHeader]**

**Cơ chế Model Binding làm việc như thế nào ?**

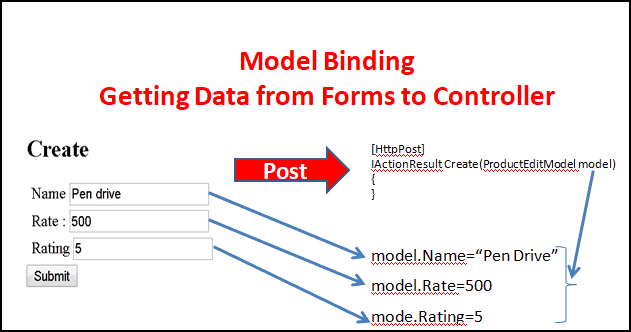
Hình dưới đây minh họa cơ chế làm việc của Model binding:



Hình 2.11: Cơ chế Model Binding

Khi người dùng click vào nút Submit thì một request Post được gửi lên server với Form Data, QueryString, Route Parameter...MVCRouteHandler của Routing Engine sẽ xử lý request đến và có trách nhiệm gọi action method tương ứng. Model Bindler sẽ được kích hoạt trước khi action method được gọi. Nó tìm dữ liệu thỏa mãn trong form data, querys tring và request parameter trong HTTP Request. Sau đso nó sẽ binding các giá trị vào tham số của action method qua tên.

Ví dụ, trường "**name**" trong form sẽ được map vào thuộc tính "**Name**" trong **ProductEditModel**. Rate trong form sẽ được map vào thuộc tính Rate...



Hình 2.12: Minh họa cơ chế Model Binding

**Model Binder:**

Model Binder có trách nhiệm gán dữ liệu vào các tham số của action method. Model Binder được tạo mởi model binder provider. Model binder phải được implement inteface **IModelBinderProvider**. Nghĩa là bạn có thể tạo một Model Binder của riêng mình hoặc mở rộng nó bằng cách triển khai inteface **IModelBinderProvider**. Custom model binder phải được đăng ký trong **ModelBinderProviders**trong **Startup.cs**.

services.AddMvc(options =>

{

options.ModelBinderProviders.Add(new CustomModelBinderProvider());

});

**ModelState:**

Nếu Model binder không thành công trong việc bind dữ liệu từ Request vào thuộc tính model tương ứng, nó sẽ không đưa ra bất cứ thông báo lỗi nào. Nhưng nó sẽ update đối tượng ModelState với danh sách lỗi và set thuộc tính IsValid là false.

Vì thế kiểm tra **ModelState.IsValid** sẽ cho chúng ta thấy quá trình binding có thành công hay không.

|  |  |
| --- | --- |
| Source | Mô tả |
| Request.Form | Chứa dữ liệu từ form html |
| RouteData.Values | Giá trị định tuyến các ứng dụng |
| Request.QueryString | Dữ liệu trong chuỗi query của ỦL |
| Request.Files | Những tệp tun được upload như một phần của yêu cầu |

Bảng 2.2: Default model binder

Ví dụ: Để tìm một tham số “id”:

* Request.Form[“id”]
* RouteData.Values[“id”]
* Request.QueryString[“id”]
* Request.Files[“id”]

**Ràng buộc những kiểu tham số đơn giản**

Khi làm việc với những kiểu tham số đơn giản, DefaultModelBinder cố gắng chuyển đổi giá trị chuỗi mà được thu thập từ dữ liệu yêu cầu sang kiểu của tham số.

Nếu giá trị không thể chuyển đổi, ví dụ như chúng ta cung cấp giá trị kiểu string cho tham số mà yêu cầu có giá trị int, DefaultModelBinder không thể rang buộc tới model.

Chúng ta có thể sửa đổi tham số để tránh gặp phải vấn đề này. Chúng ra sử dụng kiểu nullable

VD : public IActionResult register (int? id)

Nếu chúng ta sử dụng cách tiếp cận này, tham số id sẽ null nếu không khớp với dữ liệu chuyển đổi đã thấy trong yêu cầu

**Ràng buộc những kiểu tham số phức tạp**

Khi tham số của action method có kiểu phúc tạp ( những kiểu dữ liệu mà không thể chuyển đổi bằng cách sữ dụng lớp TypeConverter), lớp DefaultModelBinder sử dụng các phản chiếu để thu được tập hợp các thuộc tính public và ràng buộc lần lượt trong đó.

Default model binder sẽ kiểm tra các thuộc tính xem liệu chúng có là kiểu dữ liệu đơn giản. Nếu đúng, binder sẽ tìm mục dữ liệu ( data item ) trong yêu cầu có cùng tên với thuộc tính.

Nếu các thuộc tính là kiểu phức tạp khác, thì quá trình này lặp lại cho kiểu mới, một bộ các thuộc tính public được thu thập, và binder lại cố gắng tìm giá trị của tất cả. Sự khác biệt là tên thuộc tính được lồng nhau

##### **Model validation:**

Model validation là quá trình đảm bảo dữ liệu mà chúng ta nhận được phù hợp để rang buộc với model của chúng ta. Nó còn cung cấp những thông tin hữu ích để giúp người sử dụng khác phục được vấn đề

Phần đầu của quá trình là kiểm tra dữ liệu nhận được. Đây là một trong những cách chúng ta bảo vệ tính toàn vẹn của domain model. Bằng cách loại bỏ những dữ liệu không có ý nghĩa trong domain model, chúng ta ngăn chặn được những trạng thái kỳ lạ và không mong muốn phát sinh trong ứng dụng. Phần hai quan trongj không kém, đó là giúp đỡ người sử dụng khắc phục lỗi. Nếu chúng ta không cung cấp cho người sử dụng thông tin và công cụ để họ tương tác với ứng dụng theo cách mà chúng ta cần họ làm, sẽ dẫn đến kết quả không mong muốn

**Sự quan trọng của client-side validation**

* Giúp tăng sự trải nghiệm
* Vì việc kiểm tra tiến hành phía trình duyệt client nên phản hồi nhanh hơn và gần như là ngay lập tức
* Tiết kiệm tài nguyên server như là băng thông bằng cách giảm truy vấn đến server.

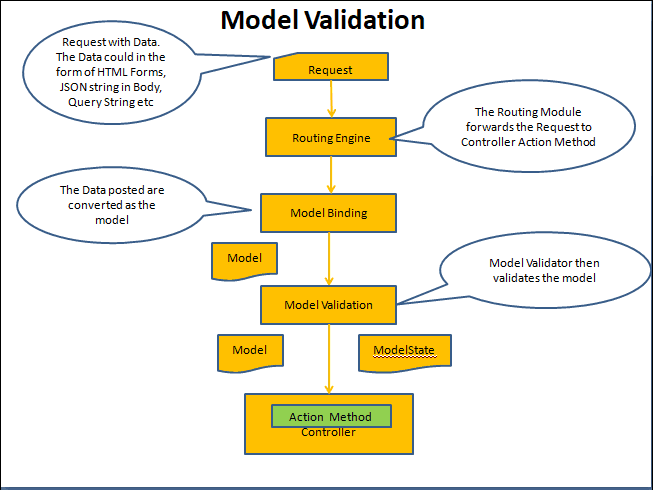
**Sự quan trọng của server-side validation**

Client-side validation cung cấp trải nghiệm người dùng tốt hơn nhưng không tin cậy. Nó có thể lỗi do một trong các lý do sau:

* Javascript có thể bị tắt ở trình duyệt
* Người dùng có thể gửi trực tiếp dữ liệu đến người dùng mà không sử dụng ứng dụng hoặc sử dụng một số các trình chỉnh sửa request có hại.
* Khi Javascript có lỗi thì kết quả là dữ liệu được đưa vào hệ thống mà có thể không hợp lệ

Vì thế điều quan trọng là kiểm tra dữ liệu phải được thực hiện ở cả phía server, ngay cả bạn đã validate ở phía client.

Tất cả các logic kiểm tra đều chạy phía server. ASP.NET Core có nhiều các attribute viết sẵn, bạn có thể thêm nó vào thuộc tính của model để cài đặt quy tắc kiểm tra. Các quy tắc này cũng được gọi là **DataAnnotations**và nằm trong namespace: **System.ComponentModel.DataAnnotations**. Bạn cần import nó vào **ViewModel**và sử dụng

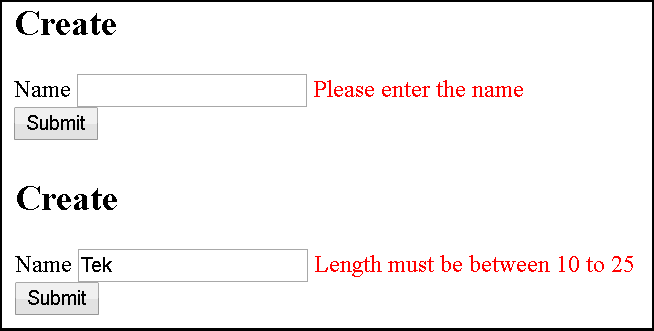
****

Hình 2.13: Mô hình hoạt động của model validation

**Hiển thị danh sách lỗi Validations**

Trong view, sử dùng Validation Tag Helper để hiển thị lỗi cho người dùng. Tag helper **asp-validation-summary** giúp hiển thị danh sách các thông báo lỗi cho riêng Model validation trên form. Nó được gắn vào thẻ div và đặt trên cùng của form. **asp-validation-for** hiển thị danh sách lỗi cho thuộc tính **Name**ở phía bên phải của nó. Nó được gắn vào thẻ span và đặt ngay cạnh thuộc tính.

Validation tag helper thêm class **field-validation-error** và **validation-summary-errors**vào HTML nếu lỗi tìm thấy. Vì thế chúng a thêm các style CSS vào để hiển thị lỗi màu đỏ cho các class này

****

Hình 2.14: Thông báo lỗi model valitaion

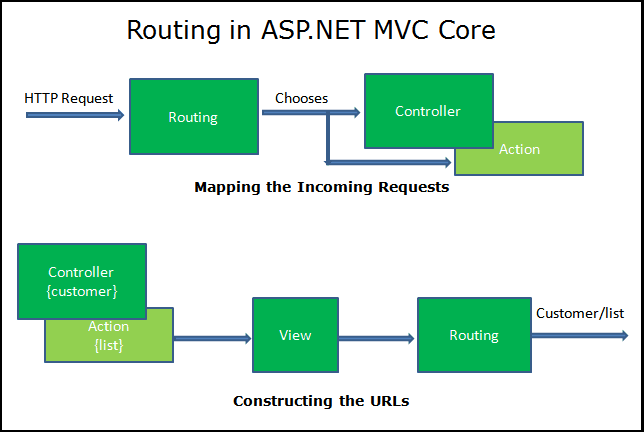
#### URL và Routing

ASP.NET Core có một bộ máy ánh xạ URl rất mạnh mẽ. Bộ máy này cung cấp phương pháp linh hoạt trong việc ánh xạ URL sang cho controller. Chúng ta có thể dễ dàng đặt ra các quy luật ánh xạ, cài đặt để ASP.NET Core MVC dựa vào các quy luật ánh xạ đó, xác định xem thực thi controller nào. ASP.NET Core còn có khả năng phân tích URL, chuyển các thông số trong URL thành các tham số trong phần gọi hàm của controller

Khi chúng ta thực hiện một yêu cầu từ ứng dụng, thì yêu cầu đó sẽ được chuyển tiếp đến một controller nào đó trong ứng dụng, và để thực hiện điều này chúng ta cần đến tính năng Routing

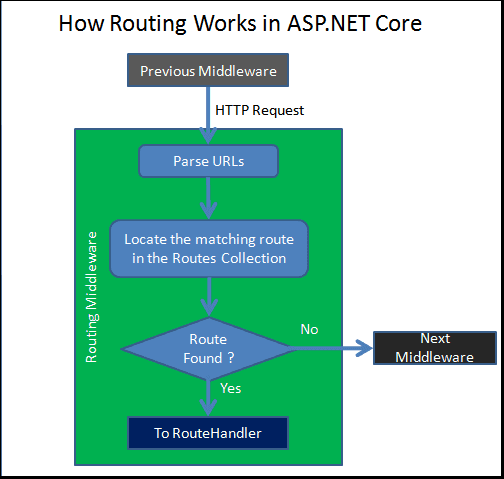
**Routing là gì?**

Routing là một quá trình khi ASP.NET Core xem xét các URL request gửi đến và "chỉ đường" cho nó đến Controller Actions. Nó cũng được sử dụng để tạo ra URL đầu ra. Quá trình này được đảm nhiệm bởi Routing Middleware. Routing Middleware có sẵn trong thư viện **Microsoft.AspNetCore.Routing**.



Hình 2.15: Routing

Hình dưới đây minh họa cách routing hoạt động:



Hình 2.16: Cách routing hoạt động

Khi request đến thì Routing Middleware sẽ làm những việc sau đây:

* Phân tích URL
* Tìm kiếm xem có cái Route nào match trong **RouteCollection**
* Nếu Route tìm thấy thì đẩy nó sang **RouteHandler**
* Nếu không tìm thấy Route nào thì bỏ qua và gọi middleware tiếp theo

**Routing là gì?**

Route tương tự như bản đồ. Chúng ta sử dụng bản đồ để đi đến điểm đích. Tương tự như thế, ứng dụng ASP.NET Core sử dụng Route để đến controller action.

Mỗi Route bao gồm các thông tin như tên, mẫu URL (URL pattern) hay còn gọi là template url, thông tin controller action mặc định và ràng buộc (constraints). URL pattern được so sánh với URL đến xem có đúng mẫu không. Một ví dụ của URL pattern là: {controller=Home}/{action=Index}/{id?}

Route được định nghĩa trong Microsoft.AspNetCore.Routing.

**Route Collection là gì?**

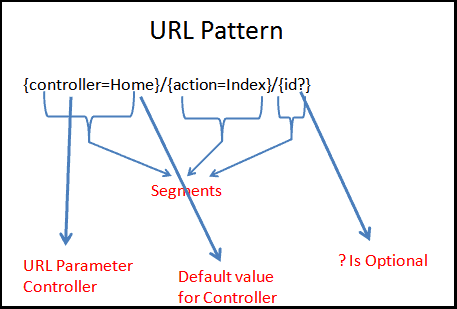
Route Collection là một tập hợp tất cả ác Route trong ứng dụng. Một ứng dụng sẽ lưu một tập hợp các route ở một nơi duy nhất trong bộ nhớ. Các Route này sẽ thêm vào collection khi ứng dụng khởi động. Route Module sẽ tìm kiếm một Route match với URL request đến trong mỗi một Route của Route Collection. Route Collection được định nghĩa trong **Microsoft.AspNetCore.Routing.**

**MVCRouteHandler là gì?**

Đây là Route Handler mặc định của ASP.NET Core MVC Middleware. MVCRouteHandler được đăng ký khi đăng ký MVC Middleware. Bạn có thể ghi đè việc này bằng cách tự tạo cho mình một custom implementation của Route Handler.

MVCRouteHandler được định nghĩa trong namespace: Microsoft.AspnetCore.Mvc MVCRouteHandler có trách nhiệm gọi Controller Factory, sau đó nó sẽ tạo ra một thể hiện của Controller được ghi trong Route. Controller sẽ được nhận và nó sẽ gọi một Action Memthod và tạo ra View. Vậy là hoàn thành request.

**URL Patterns**

****

Hình 2.17: URL pattern

Mỗi Route phải chứa một URL Pattern. Pattern này sẽ được so sánh với URL requét. Nếu pattern đúng với URL thì nó sẽ được sử dụng bởi hệ thống routing để xử lý URL đó.

Mỗi một URL Pattern bao gồm một hoặc nhiều phần. Các phần chia tách bởi dấu gạch chéo. Mỗi phần có thể là một hằng số (constant) hoặc một Route Parameter.

Route Parameter được bao gọc bởi một cặp dấu ngoặc nhọn ví dụ {controller}, {action}. Route Parameter có thể có giá trị mặc định như {controller=Home} khi Home là giá trị mặc định của controller. Một dấu = sẽ gán giá trị cho tên parameter.

Bạn có thể có một thành phần dạng hằng số. Ví dụ: admin/{controller=Home}/{action=Index}/{id?}. Ở đây thì "admin" là một hằng tức là một chuỗi cố định phải tồn tại trên URL.

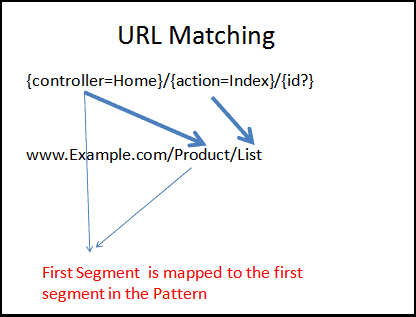
Dấu ? trong {id?} chỉ ra là tham số này không bắt buộc. Một dấu ? sau tên tham số chỉ ra tham số đó không yêu cầu phải có giá trị.

URL Pattern {controller=Home}/{action=Index}/{id?}. Đăng ký một route có thành phần đầu tiên trên URL là một controller, phần thứ 2 là Action method trong controller đó. Và phần cuối là dữ liệu thêm vào tên là id.

**URL Matching**

Mỗi phần trong URL request đến sẽ match tương ứng với thành phần của URL Pattern. Route {controller=Home}/{action=Index}/{id?} có 3 thành phần. Phần cuối là tùy chọn. Xem xét ví dụ URL www.example.com/product/list thì URL này có 2 thành phần. URL này vẫn match với pattern ở trên vì phần thứ 3 không yêu cầu.

Routing Engine sẽ nhận diện {controller}= Product & {action}= List

****

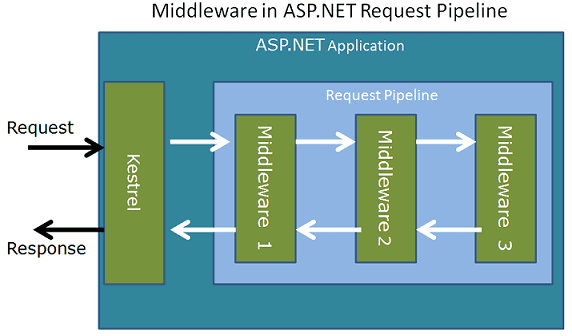
Hình 2.18: Url Matching

#### Middleware

**Middleware là gì ?**

Middleware là thành phần của phần mềm đóng vai trò tác động vào request pipeline (luồng request) để xử lý chúng và tạo ra response phản hồi lại client. Mỗi một tiến trình middleware thao tác với các request nhận được từ middleware trước nó. Nó cũng có thể quyết định gọi middleware tiếp theo trong pipeline hoặc trả về response cho middleware ngay trước nó. (ngắt pipeline).

**Cách middleware hoạt động :**

****

Hình 2.19: Cách middleware hoạt động

Đầu tiên, HTTP Request đến (trực tiếp hoặc qua External web server) ứng dụng. Kestrel web server nhặt lấy request và tạo một HttpContext và gán nó vào Middleware đầu tiên trong request pipeline.

Middleware đầu tiên sẽ nhận request, xử lý và gán nó cho middleware tiếp theo. Quá trình này tiếp diễn cho đến khi đi đến middleware cuối cùng. Tùy thuộc bạn muốn pipeline của bạn có bao nhiêu middleware.

Middleware cuối cùng sẽ trả request ngược lại cho middleware trước đó, và sẽ ngắt quá trình trong request pipeline.

Mỗi Middleware trong pipeline sẽ tuần tự có cơ hội thứ hai để kiểm tra lại request và điểm chỉnh response trước khi được trả lại.

Cuối cùng, response sẽ đến Kestrel nó sẽ trả response về cho client. Bất cứ middleware nào trong request pipeline đều có thể ngắt request pipeline tại chỗ đó với chỉ một bước đơn giản là không gán request đó đi tiếp.

Có 3 loại middleware, phân loại bằng cách bạn implement nó như nào

| **Loại** | **Có thể short-circuit** | **Dùng để** |
| --- | --- | --- |
| Use | Có | Short-circuit một request Logic để tạo response |
| Run | Không | Kết thúc pipeline |
| Map MapWhen | Không | Phân nhánh pipeline dựa trên request path MapWhen phân nhánh dựa trên điều kiện Hỗ trợ Nesting (multi-level branching) |

Bảng 2.3 Các loại middleware

#### Cơ chế Dependency Injection

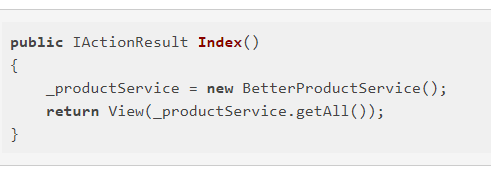
**Giới thiệu về Dependency Injection**

Dependency Injection giờ đã trở thành thành phần chính thức mặc định của ASP.NET Core. Nó giúp chúng ta đáp ứng tính chất lỏng lẻo (loosely couple), dễ đọc và bảo trì code. Trong bài viết này chúng ta sẽ học cơ bản về Dependency Injection trong việc xây dựng ứng dụng đơn giản

**Dependency Injection là gì ?**

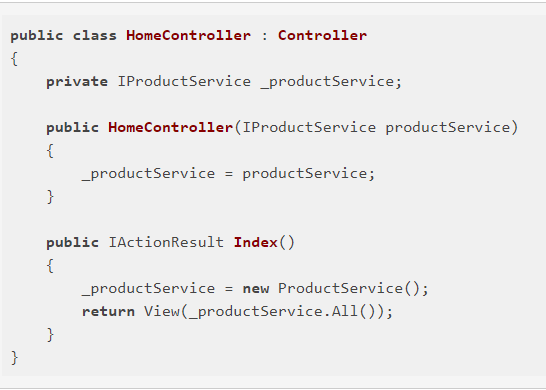
Dependency Injection (được biệt là DI) là một design pattern khi một đối tượng không được tạo trong các thành phần phụ thuộc vào nó mà yêu cầu nó

ví dụ sau đây trong controller action method khi chúng ta muốn lấy danh sách sản phẩm ra từ **ProductService**.

****

Hình 2.20: Method không sử dụng DI

Action method **Index**có một phụ thuộc đến **ProductService**. Vì thế nó tạo ra một thể hiện và gọi phương thức phương thức **GetAll**để lấy danh sách sản phẩm. **ProductService**giờ đã bị gắn chặt (tightly couple) vào phương thức **Index**của **HomeController**



Hình 2.21: Moethod sử dụng DI

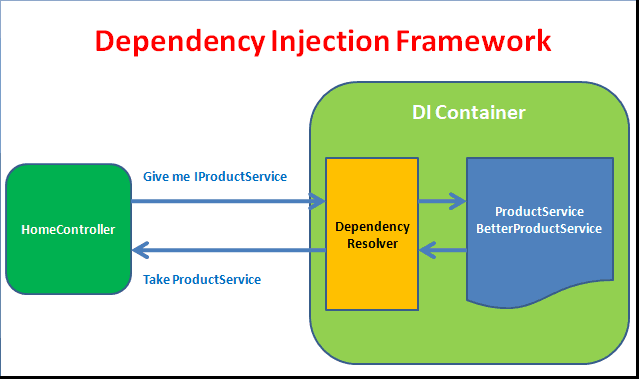
Đây là điểm mà ASP.NET Core Dependency Injection framework làm nhiệm vụ của nó. Trách nhiệm của nó là tạo ra thể hiện của ProductService và đối tượng này được gọi là **DI Container** hay **Ioc Container.**

Dependency Injection là một design pattern. Dependency injection framework triển khai design pattern này. Có nhiều framework như Autofac, Unity...bạn có thể sử dụng trong ASP.NET Core.

**DI Container:**

DI Container là một đối tượng có trách nhiệm tạo các phụ thuộc (**ProductService**) và gán nó cho đối tượng yêu cầu (**HomeController**) nó.

**Cách hoạt động của Dependency Injection**

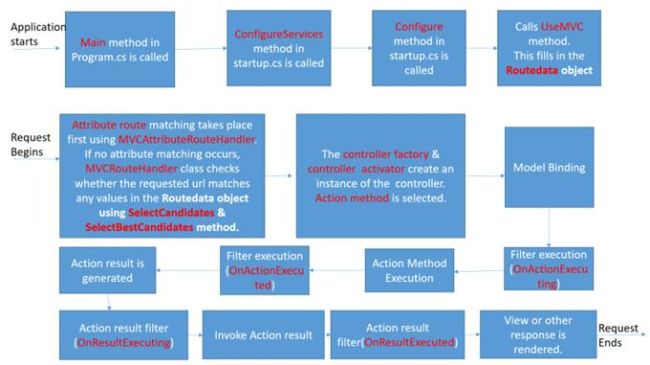
****

Hình 2.22: Cách hoạt động của dependency injection

**Lợi ích của Dependency Injection**

* Dependency injection giúp thực hiện kiến trúc lỏng lẻo (loose coupling) trong phần mềm.
* Code sẽ sạch và dễ đọc hơn
* Tăng khả năng có thể kiểm thử và bảo trì
* Cho phép bạn thay đổi triển khai mà không phải thay đổi quá nhiều code.

### Quá trình hoạt động của một ứng dụng ASP.NET Core

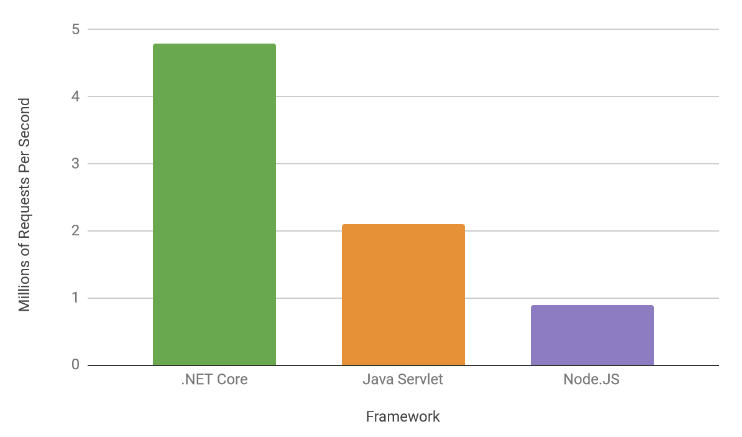


Hình 2.23: Vòng đời của ứng dụng asp.net core

### Ưu điểm và nhược điểm

**Ưu điểm:**

* ASP.NET Core có một số thay đổi kiến trúc dẫn đến modular framework nhỏ hơn.
* ASP.NET Core không còn dựa trên System.Web.dll. Nó dựa trên một tập hợp nhiều yếu tố của Nuget packages.
* Điều này cho phép bạn tối ưu ứng dụng của mình chỉ cần những NuGet packages cần thiết.
* Lợi ích của diện tích bề mặt ứng dụng nhỏ hơn thì bảo mật chặt chẽ hơn, giảm dịch vụ, cải thiện hiệu suất và giảm chi phí.
* Xây dựng và chạy các ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.
* Được xây dựng trên **.NET Core** hỗ trợ side-by-side app versioning.
* Công cụ mới giúp đơn giản hóa việc phát triển web hiện đại.
* Liên kết đơn các web stack như Web UI và API Web.
* Cấu hình dựa trên môi trường đám mây sẵn có.
* Được xây dựng dựa trên cho DI (Dependency Injection).
* Tag Helpers làm cho các Razor makup trở nên tự nhiên hơn với HTML.
* Có khả năng host trên IIS hoặc self-host



Hình 2.24: So sánh số lượng Request trong 1 giây

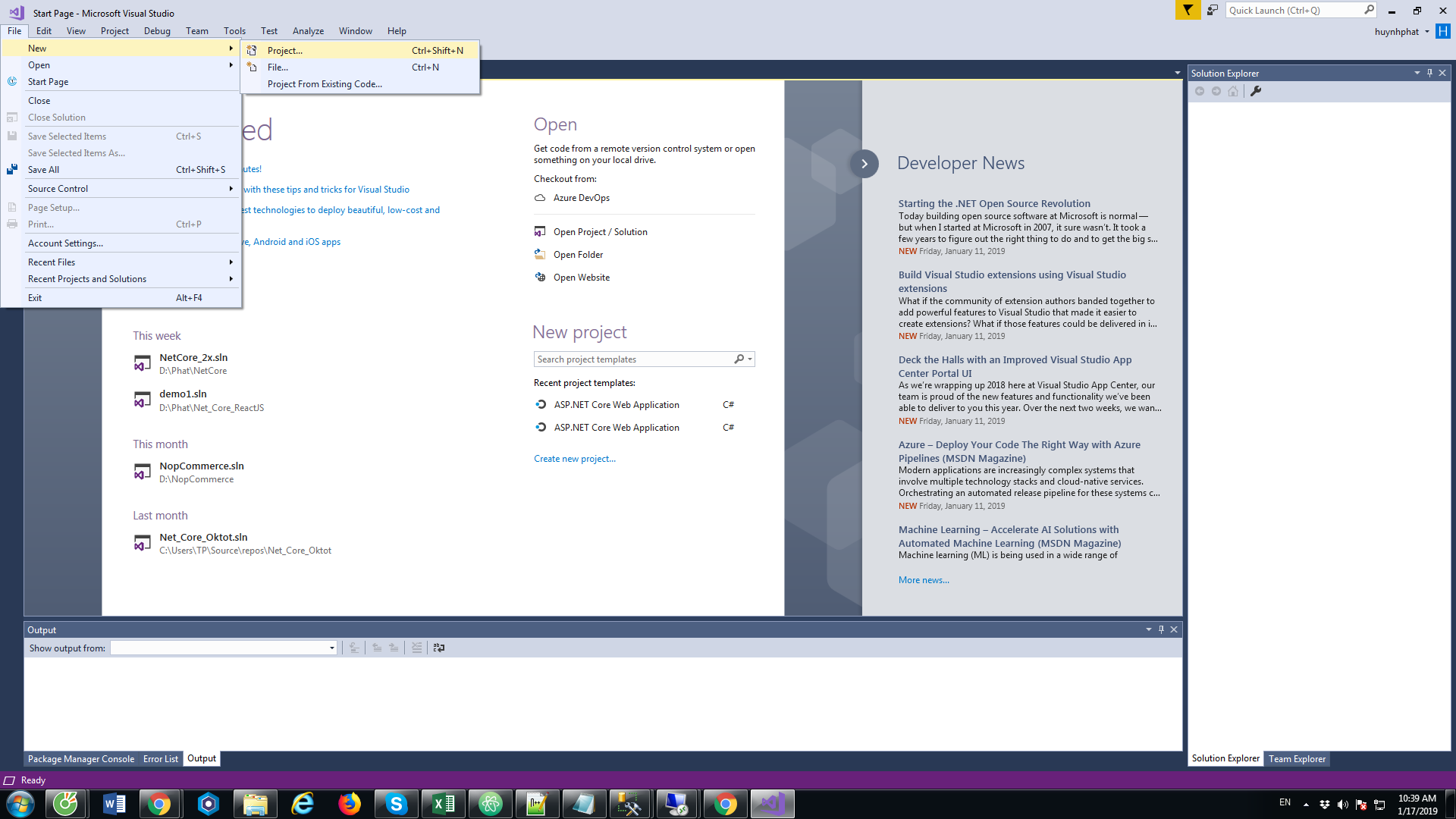
**Nhược điểm:**

* Không phái tất cả thư viện đều hỗ trợ asp.net core
* Công cụ phát triển dựa trên Windows

### Làm quen với ứng dụng ASP.NET Core:

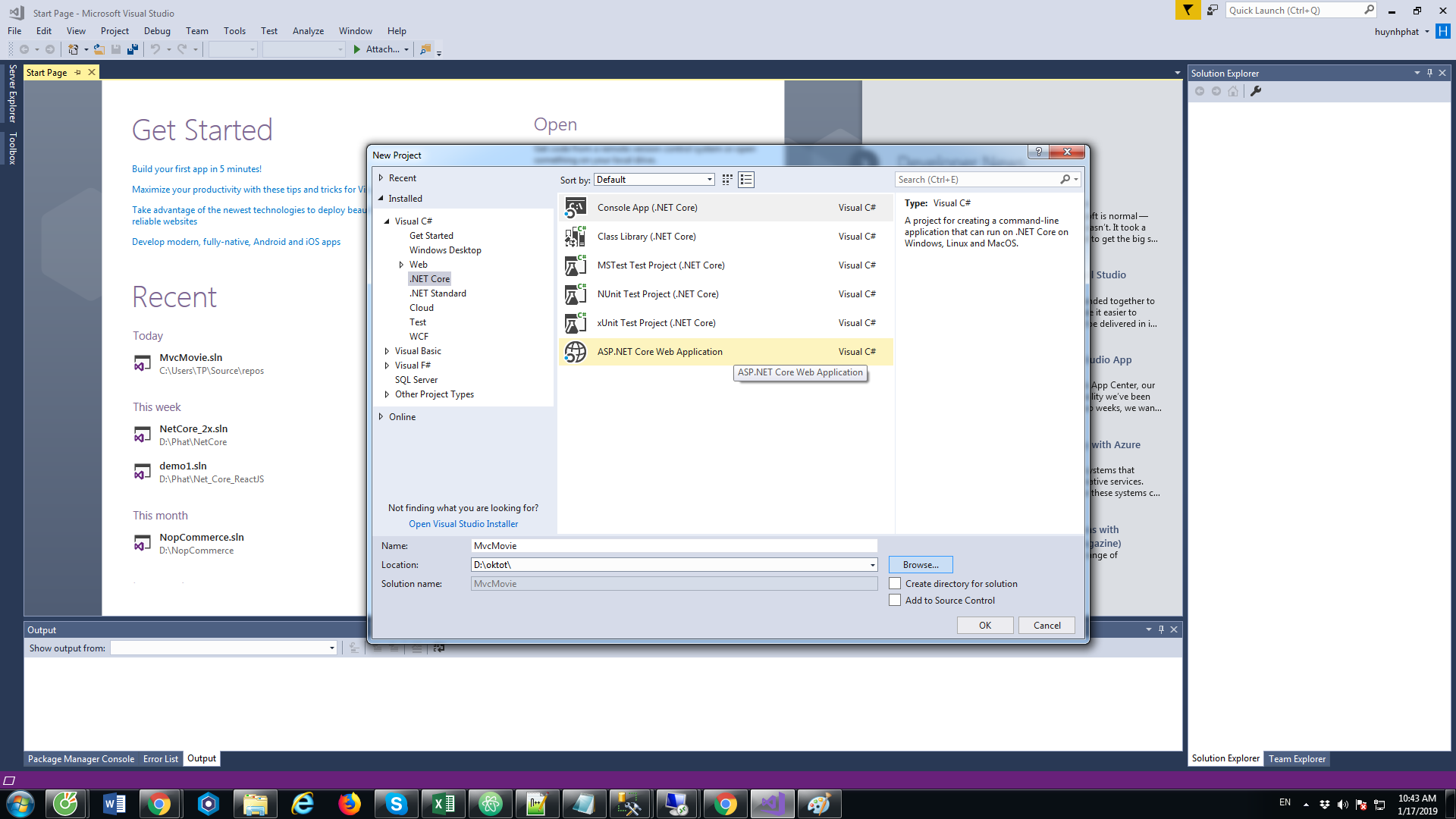
#### Cách khởi tạo dự án ASP.NET Core:

Trên Visual Studio nhấn **File > New > Project**.



Hình 2.25: Mở hộp thoại tạo project ASP.NET Core ( 1 )

Trong ngăn trung tâm, chọn Ứng dụng web ASP.NET Core (.NET Core). Đặt tên cho dự án

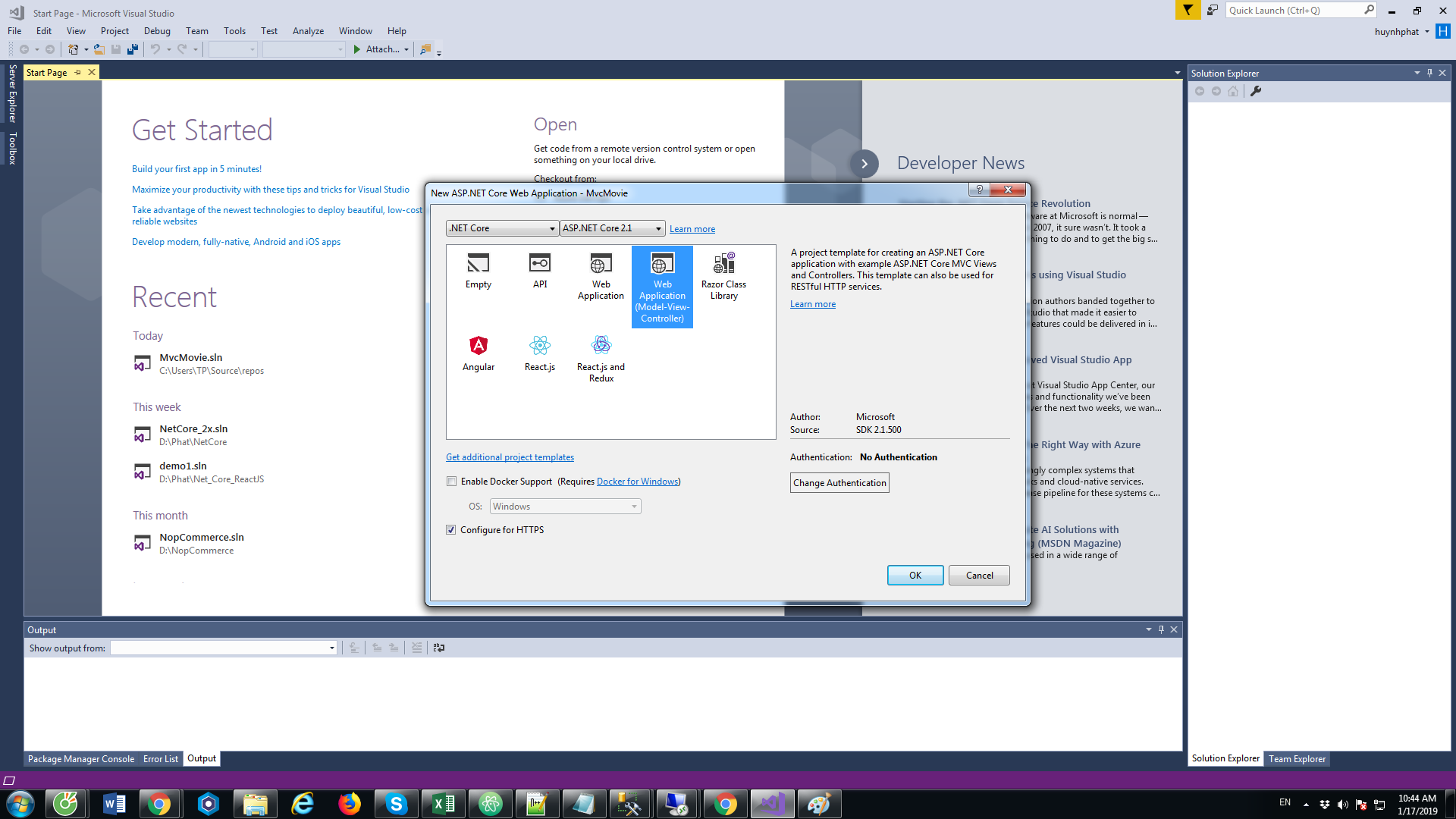


Hình 2.26: Đặt tên project

Chúng ta có thể tùy chọn phiên bản .net core và chọn kiểu template:

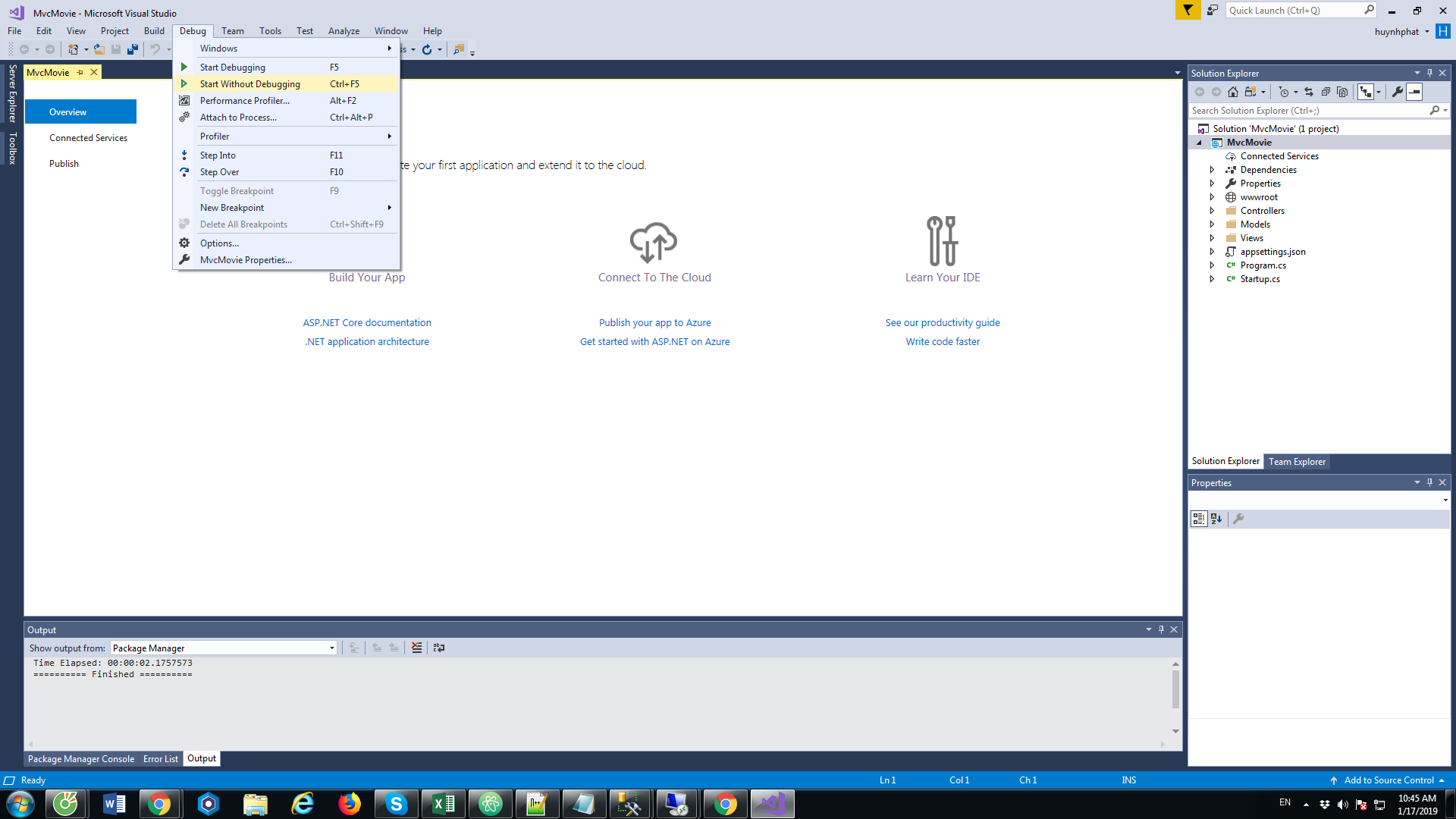
* Empty sẽ tạo những thư mục và tập tin thối thiểu cho ứng dụng
* API sẽ tạo ứng dụng dưới dạng web service cung cấp các dịch vụ để truy xuất, thay đổi dữ liệu
* Web application tạo project web
* Web application ( MVC ) tạo project phát triển web áp dụng mô hình MVC
* ASP.NET Core có thể tích hợp them các framework fontend của javascript như React.js , Angular,...

Chúng ta có thể tùy chọn sử dụng HTTP và HTTPS và các phương thức xác thực được cung cấp sẵn



Hình 2.27: Mô hình MVC trong project

Chạy project ASP.NET Core



Hình 2.28: Chạy project

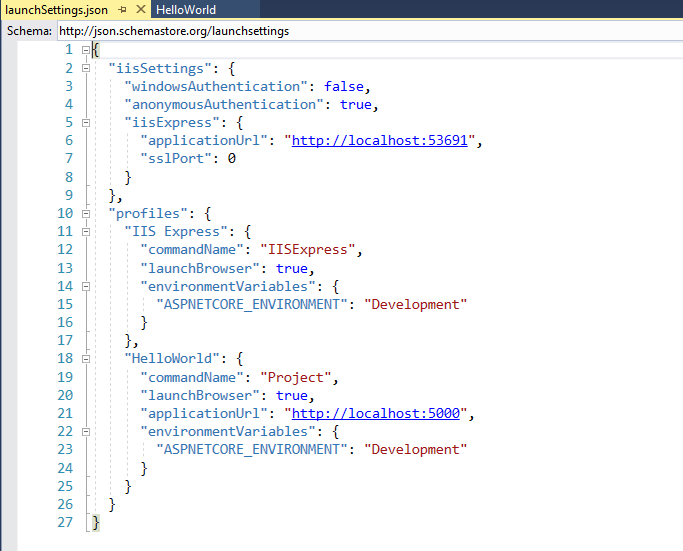
#### Cấu trúc dự án:

Hình .: Cấu trúc của một dự án ASP.NET Core

Solution chứa 3 folder là Dependencies, Properties và wwwroot. Thư mục wwwroot ở đây không được khởi tao mặc định nhưng chúng ta có thể thêm mới bằng tay. Ngoài ra nó còn chứa 2 file rất quan trọng là Program.cs và Startup.cs.

**Thư mục Properties:**

Thư mục Properties chứa một file gọi là **launchSettings.json**. File json này chứa tất cả thông tin cụ thể cài đặt project để có thể chạy được ứng dụng. Bạn sẽ thấy các profile debug, các biến môi trường được chỉ ra trong file này.

****

Hình 2.30: Cấu trúc launchSetting

**Dependencies:**

Thư mục này chứa toàn bộ các thành phần phụ thuộc dùng cho ứng dụng, chúng ta gọi là Dependencies. Visual Studio sử dụng Nuget Packages để quản lý tất cả các dependencies phía server. Với Client tức là các thư viện JS thì chúng ta sử dụng Client-side Libraries. Đây là sự khác biệt so với phiên bản trước khi mà Nuget quản lý cả dependencies phía server và client.

Trong thư mục Dependencies, chúng ta có thư mục Nuget chứa các Nuget Packages đang sử dụng và thư mục SDK chứa Microsoft.NETCore.App. Đây là .NET Core Runtime sẽ sử dụng trong project

**Các file hệ thống:**

File project (.csproj) đã có nhiều thay đổi từ Visual Studio 2017. Nó đã đơn giản rất nhiều rồi. Có một sự tương quan trưc jtieeps giữa các file trong thư mục solution và những gì hiển thị trong Solution Explorer của Visual Studio. Các file bạn add vào solution folder nó cũng tự động trở thành một phần của project.

Bảng bưới đây mô tả chức năng của các file sau khi tạo project:

|  |  |
| --- | --- |
| File | Mô tả |
| ClientApp | Thư mục chứa tất cả file giao diện ( html ), xử lý ( ts ) , file cấu hình của Angular ( ứng dụng single page ) |
| Areas | Dùng để chia ứng dụng thành từng vùng tương ứng cho các vai trò khác nhau |
| Controller | Chứa tất cả controller của ứng dụng |
| Model | Chứa các xử lý logic , class đối tượng, kết nối database của ứng dụng |
| Views | Chứa tất cả các file giao diện ( cshtml ) của ừng dụng ( nếu không dùng single page ) |
| Appsettings | Chứa các biến lưu giá trị dung chung cho ứng dụng |
| Startup.cs | Chứa tất cả cấu hình của ứng dụng ASP.NET Core |

Bảng 2.4: chức năng của các file sau khi tạo project

# CHƯƠNG TRÌNH ỨNG DỤNG

## Mở đầu:

Chương này sẽ đi xây vào xác định, phân tích chi tiết và xây dựng các chức năng của hệ thống

Hệ thống sẽ bao gồm các chức năng sau:

|  |  |  |
| --- | --- | --- |
| STT | Tên Usecase | Ý nghĩa/Ghi chú |
| 1 | Đăng nhập |  |
| 2 | Đăng xuất |  |
| 3 | Đăng ký |  |
| 4 | Danh sách đề thi | * Xem danh sách đề thi theo danh mục ( tích phân, đồ thị, hình học,…) hoặc theo độ khó ( dễ , vừa , khó ) * xem chi tiết ( nội dung chi tiết các câu hỏi , các câu trả lời của đề thi ) * tải đề thi |
| 5 | Làm bài thi toán online |  |
| 6 | Tìm kiếm | * tìm kiếm dựa theo tên , độ khó , loại câu hỏi |
| 7 | Xem điểm và nội dung bài làm của các đề đã thi | * Danh sách các bài thi đã làm gồm ( điểm số từng bài , chi tiết các đáp án đã chọn và đáp án đúng ) * Biểu đồ thống kê các câu trả lời đúng sai theo từng mục ( tích phân, đồ thị, hình học,…) dựa theo các bài kiểm tra đã làm trước đây * Có thể làm lại các bài thi điểm chưa tốt * Dựa trên kết quả thống kê đánh giá và đưa ra lời khuyên, bài tập cải thiện |
| 8 | Quản lý tài khoản và thông tin người dùng | Xóa / Cập nhật thông tin người dùng |
| 9 | Quản lý đề thi | * Thêm / xóa đề thi . Cập nhật thông tin chung đề thi * Tạo đề thi tự động dựa trên các thông tin ( số lượng câu hỏi cho từng phần của đề thi, độ khó, thời gian làm bài,…) * thêm/ xóa câu hỏi của đề thi |
| 10 | Quản lý câu hỏi | Thêm / Xóa / Sửa Nội dung câu hỏi |

Bảng 3.1: Danh sách các tính năng

## Xây dựng chức năng:

### Danh sách các actor

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa/Ghi chú |
| 1 | Người quản trị | Người có vị trí cao nhất, chịu trách nhiệm quản lý điều hành website. |
| 2 | Người dùng | Đăng nhập và sử dụng các chức năng đã được phân quyền trong hệ thống |

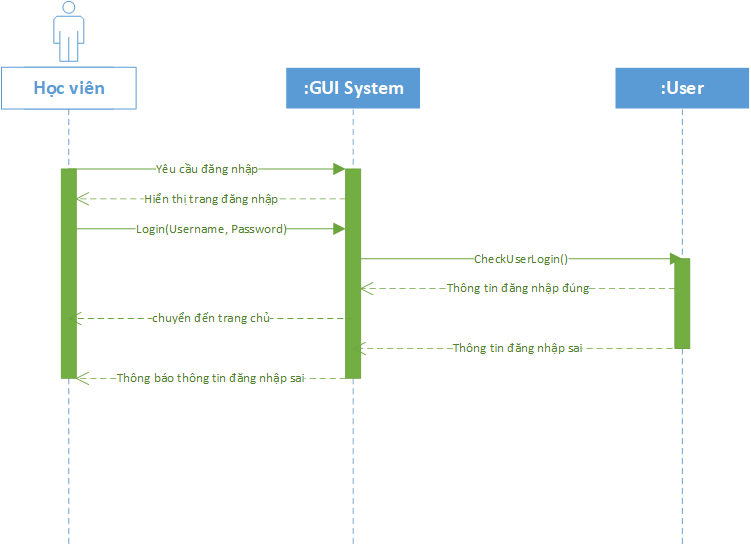
Bảng 3.2: Danh sách các actor

### Phân tích chức năng trang chính:

#### Đăng nhập:

|  |  |
| --- | --- |
| Tên chức năng | Đăng nhập |
| Tóm tắt | Chức năng đăng nhập vào hệ thống. |
| Dòng sự kiện chính | 1. Hệ thống hiển thị trang đăng nhập.  2. Người dung nhập tên và tài khoản (cả 2 trường này đều bắt buộc nhập) và nhấn “Đăng nhập”.  3. Hệ thống kiểm tra thông tin đăng nhập  (Dòng sự kiện khác: Thông tin đăng nhập sai).  4. Hệ thống hiển thị form chính. |
| Dòng sự kiện khác | 1. Thông tin đăng nhập sai:  Hệ thống hiển thị thông báo tài khoản đăng nhập không hợp lệ. |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: không có |
| Trạng thái hệ thống sau khi thực hiện use case | Người dùng đăng nhập thành công vào hệ thống, có thể sử dụng các quyền mà hệ thống cho phép |
| Điểm mở rộng | Không có |

Bảng 3.3: Dăng nhập



Hình 3.1: sequence diagram đăng nhập

#### Đăng xuất

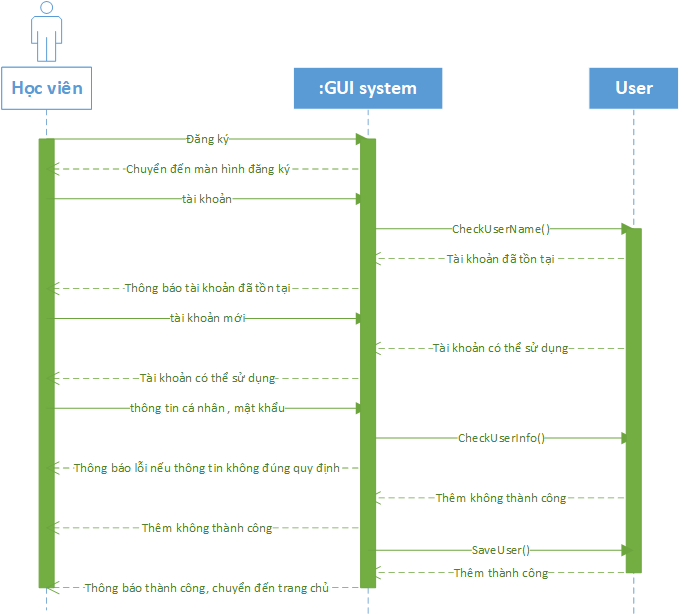
|  |  |
| --- | --- |
| Tên chức năng | Đăng xuất |
| Tóm tắt | Chức năng đăng xuất ra khỏi hệ thống |
| Dòng sự kiện chính | 1. Người dùng nhấn vào “Đăng xuất”  2. Hệ thống đăng xuất và trở về trang chủ |
| Dòng sự kiện khác | Không có |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực hiện use case | Actor: Học sinh  Điều kiện: người dùng đã đăng nhập hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Người dùng đăng xuất khỏi hệ thống |
| Điểm mở rộng | Không có |

Bảng 3.4: Đăng xuất

#### Đăng ký

|  |  |
| --- | --- |
| Tên chức năng | Đăng ký |
| Tóm tắt | Chức năng đăng ký tài khoản Người dùng. Người dùng dùng tài khoản đã đăng ký để đăng nhập và sử dụng các chức năng có yêu cầu đăng nhập của hệ thống. |
| Dòng sự kiện chính | 1. Người dùng nhấn vào nút “đăng ký” trong trang chủ  2. Hệ thống sẽ chuyển sang trang đăng ký  3. Người dùng nhập đầy đủ và đúng theo quy định các thông tin cá nhân và thông tin tài khoản mật khẩu  ( Dòng sự kiện khác :   1. thông tin đăng nhập không đúng quy định 2. Tài khoản đã tồn tại )   4. Sau khi đã nhập đúng, Người dùng chọn nút “đăng ký” |
| Dòng sự kiện khác | 1. thông tin đăng nhập không đúng quy định: hiện thông báo lỗi tại các ô thông tin sai quy định 2. Tài khoản đã tồn tại : hiện thông báo lỗi |
| Các yêu cầu đặc biệt | * Tài khoản : ít nhất 10 ký tự , ít nhật 1 kí tự viết hoa , tối da 25 ký tự * Email : đúng định dạng email * Mật khẩu: ít nhất 8 ký tự, không chứa các ký tự đặc biệt * Họ, tên: tối đa 10 ký tự * Ngày tháng năm sinh : Đúng định dạng ngày |
| Trạng thái hệ thống trước khi thực hiện use case | Actor: Học sinh |
| Trạng thái hệ thống sau khi thực hiện use case | Người dùng đăng ký thành công tài khoản |
| Điểm mở rộng | Không có |

Bảng 3.5: Đăng ký

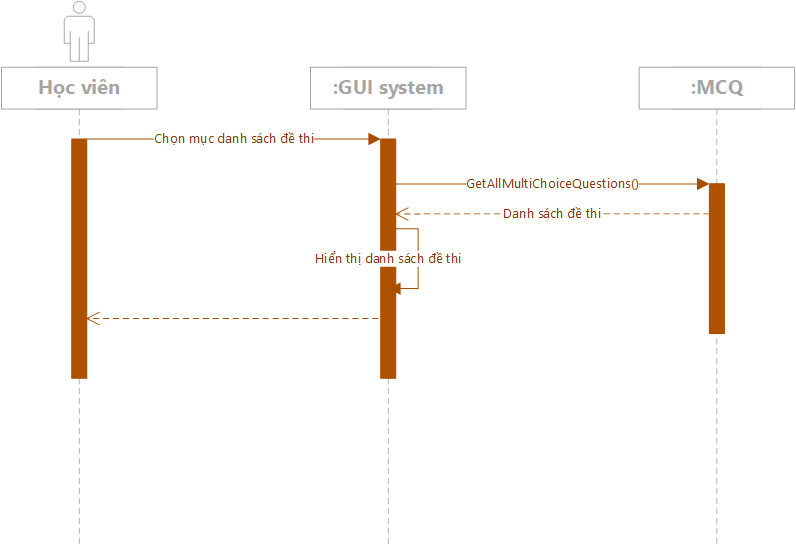


Hình 3.2: sequence diagram đăng ký

#### Xem Danh sách đề thi

|  |  |
| --- | --- |
| Tên chức năng | Xem danh sách đề thi |
| Tóm tắt | Chức năng xem danh sách toàn bộ đề thi |
| Dòng sự kiện chính | 1. Học sinh vào trang chủ hệ thống 2. Chọn mục danh sách đề thi 3. Hệ thống sẽ hiển thị danh sách toàn bộ đề thi đã có |
| Dòng sự kiện khác |  |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: không có |
| Trạng thái hệ thống sau khi thực hiện use case | Người dùng xem được danh sách đề thi |
| Điểm mở rộng | Không có |

Bảng 3.6: Xem danh sách đề thi

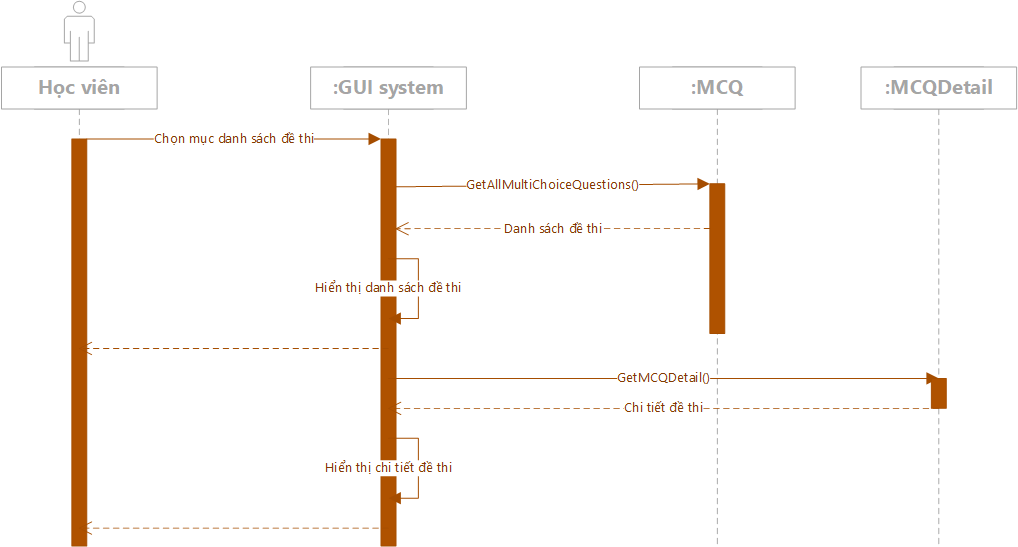


Hình 3.3: Sequence diagram xem danh sách đề thi

#### Xem chi tiết đề thi

|  |  |
| --- | --- |
| Tên chức năng | Xem nội dung chi tiết của đề thi |
| Tóm tắt | Chức năng xem các quy định, thông tin chung và nội dung chi tiết của đề thi |
| Dòng sự kiện chính | 1. Học sinh vào trang chủ và chọn mục danh sách đề thi 2. Hệ thống sẽ hiển thị danh sách toàn bộ đề thi 3. Học sinh chọn đề thi cụ thể 4. Hệ thống sẽ hiển thị trang chi tiết đề thi |
| Dòng sự kiện khác | Không có |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: không có |
| Trạng thái hệ thống sau khi thực hiện use case | Người dùng xem được các quy định, thông tin chung và nội dung chi tiết của đề thi |
| Điểm mở rộng | Không có |

Bảng 3.7: Xem chi tiết đề thi

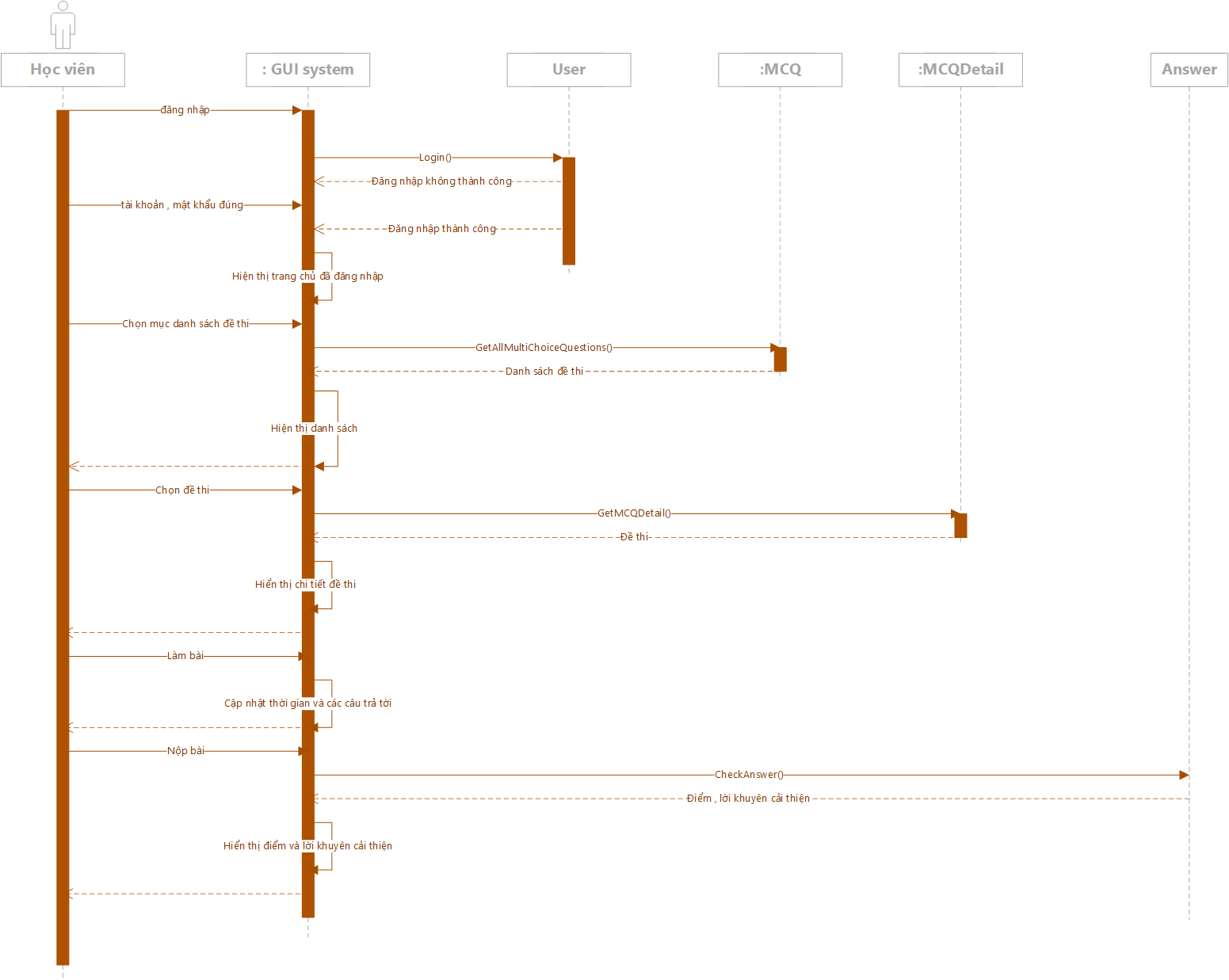


Hình 3.4: Sequence diagram xem chi tiết đề thi

#### Làm bài thi toán online

|  |  |
| --- | --- |
| Tên chức năng | Làm bài thi toán Online |
| Tóm tắt | Chức năng cho phép người dùng làm bài thi online |
| Dòng sự kiện chính | 1. Học sinh vào trang chủ đăng nhập và chọn mục “danh sách đề thi” ( dòng sự kiện khác : đăng nhập không thành công ) 2. Hệ thống sẽ hiển thị danh sách toàn bộ đề thi 3. Học sinh chọn đề thi cụ thể 4. Hệ thống sẽ hiển thị trang chi tiết đề thi 5. Học sinh chọn “làm bài thi” 6. Hệ thống chuyển sang trang làm bài thi gồm : nội dung bài thi , các đáp án , nội quy và thời gian 7. Học sinh chọn các câu trả lời trong thời gian quy định 8. Nhấn nút “nộp bài” sau khi hoàn thành bài thi 9. Hệ thống sẽ chấm điểm. Sau đó hiển thị điểm và những lời khuyên cải thiện cho học sinh |
| Dòng sự kiện khác | Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: Học sinh đã đăng nhập vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Học sinh nhận được điểm và lời khuyên của bài thi |
| Điểm mở rộng | Không có |

Bảng 3.8: Làm bài thi toán online

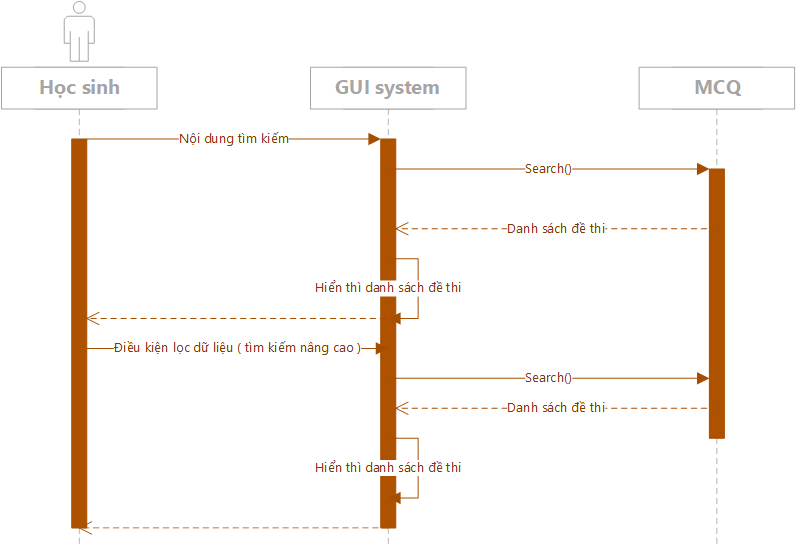


Hình 3.5: Sequence diagram làm bài thi toán

#### Tìm kiếm

|  |  |
| --- | --- |
| Tên chức năng | Tìm kiếm |
| Tóm tắt | Chức năng cho phép người dùng tìm kiếm và lọc đề thi |
| Dòng sự kiện chính | 1. Học sinh vào trang chủ 2. nhập thông tin tìm kiếm vào” thanh tìm kiếm” hoặc chọn tìm kiếm nâng cao để lọc dữ liệu tìm kiếm và nhấn nút “Tìm kiếm” 3. Hệ thống sẽ trả về danh sách đề thi theo điều kiện tìm kiếm của học sinh |
| Dòng sự kiện khác | Không có |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: không có |
| Trạng thái hệ thống sau khi thực hiện use case | Danh sách đề thi |
| Điểm mở rộng | Không có |

Bảng 3.9: TÌm kiếm

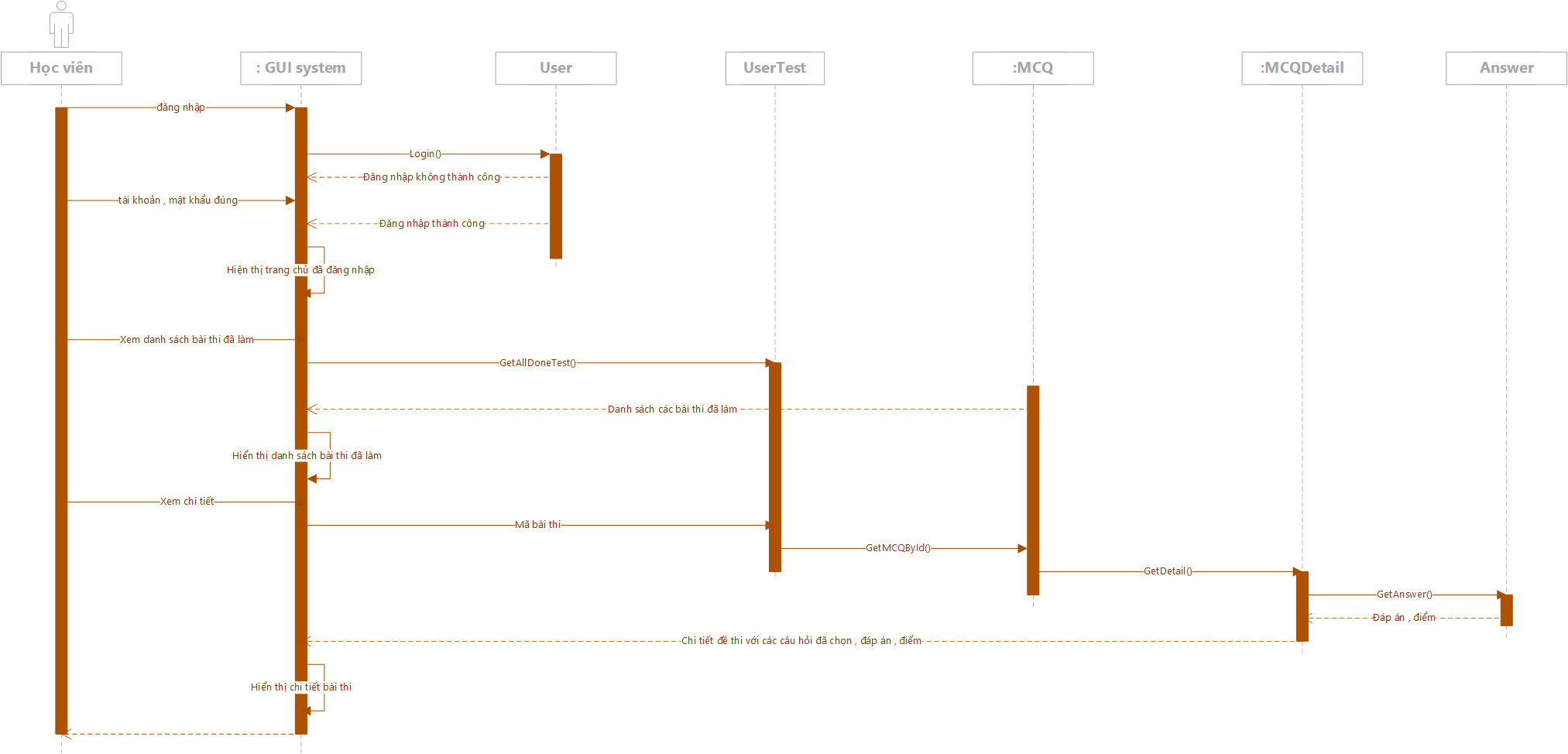


Hình 3.6: Sequence diagram tìm kiếm

#### Xem điểm và nội dung bài làm của các đề thi

|  |  |
| --- | --- |
| Tên chức năng | Xem điểm và nội dung các bài thi đã thi |
| Tóm tắt | Chức năng cho phép người xem lại điểm số , nội dung ( các đáp án đã chọn , đáp án đúng ) , lời khuyên cải thiện của các đề thi đã thi |
| Dòng sự kiện chính | 1. Học sinh vào trang chủ 2. Dùng tài khoản đã đăng ký đăng nhập vào hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 3. Chọn vào avatar học sinh trên thanh công cụ 4. Hệ thống sẽ hiện thị danh sách các chức năng 5. Học sinh chọn chức năng “Xem điểm” 6. Hệ thống sẽ hiển thị trang danh sách bài thi đã làm 7. Học sinh nhấn nút “Xem” tương ứng với bài thi cần xem 8. Hệ thống sẽ hiển thị chi tiết bài thi ( điểm số , đáp án đã chọn , đáp án đúng ) của bài thi |
| Dòng sự kiện khác | Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Học sinh  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Chi tiết bài thi |
| Điểm mở rộng | Không có |

Bảng 3.10: Xem điểm và nội dung bài làm của các đề thi đã thi



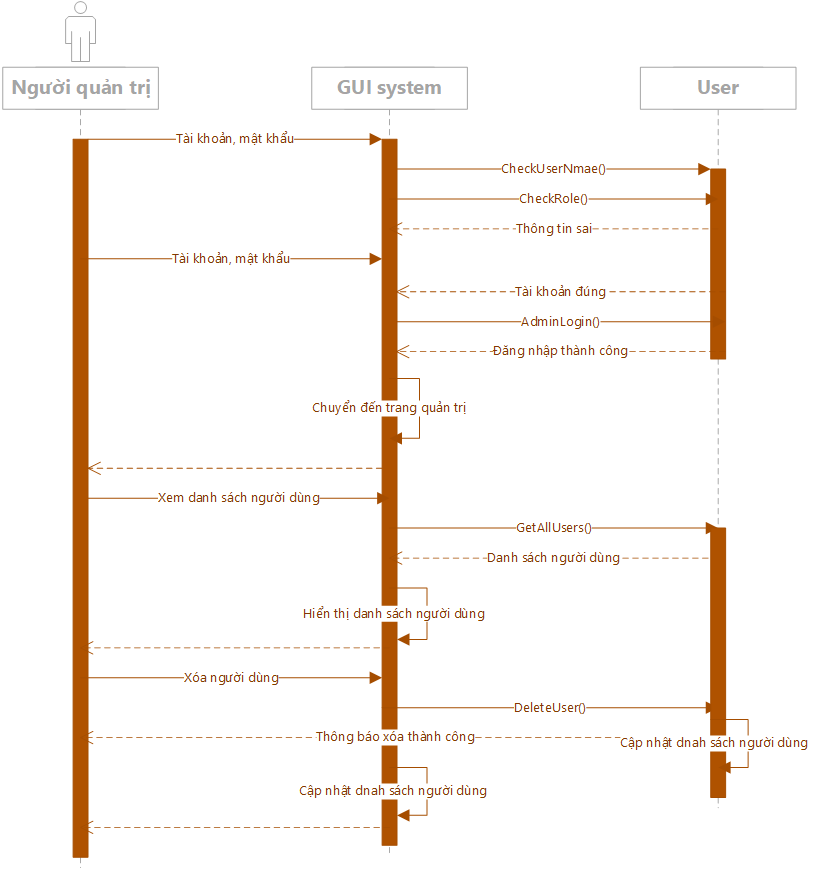
Hình 3.7: Xem điểm bài thi

### Trang admin

#### Xóa người dung

|  |  |
| --- | --- |
| Tên chức năng | Xóa người dùng |
| Tóm tắt | Chức năng cho phép người quản trị xóa thông tin học sinh |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý người dùng” 4. Hệ thống sẽ hiện thị danh sách người dùng hiện tại của hệ thống 5. Xác định học sinh cần xóa và chọn nút “Xóa” tương ứng với học sinh đó   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công học sinh. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Thông tin học sinh đã được xóa khỏi hệ thống |
| Điểm mở rộng | Không có |

Bảng 3.11 Xóa người dung

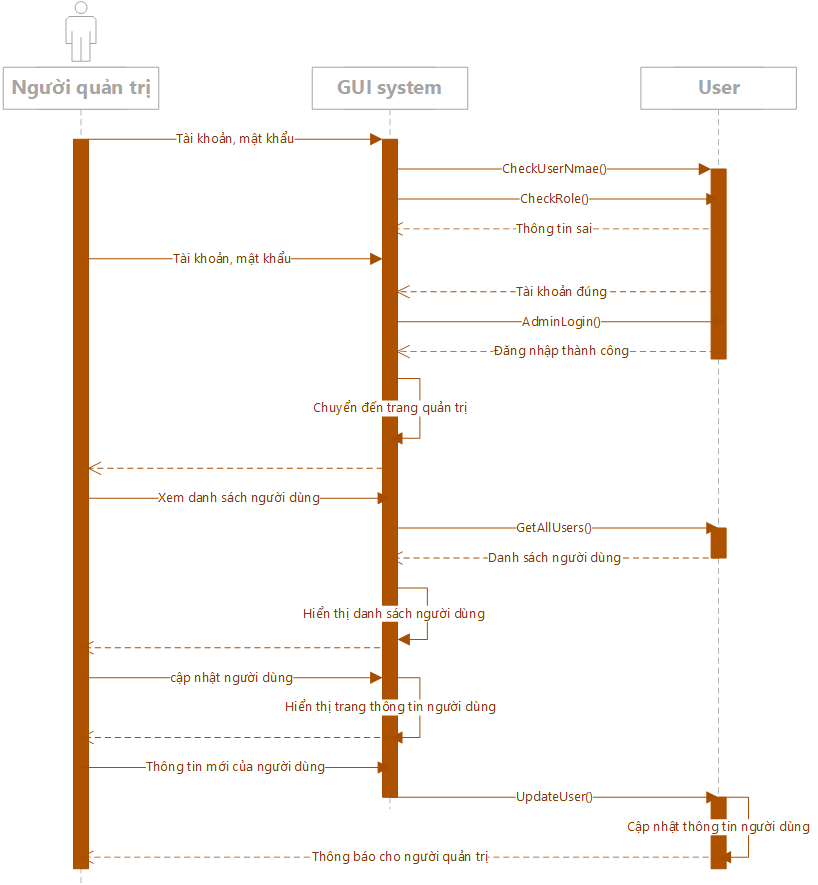


Hình 3.8: Sequence diagram xóa người dung

#### Cập nhật thông tin người dung

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin người dùng |
| Tóm tắt | Chức năng cho phép người quản trị sửa thông tin học sinh |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý người dùng” 4. Hệ thống sẽ hiện thị danh sách người dùng hiện tại của hệ thống 5. Xác định học sinh cần sửa và chọn nút “Sửa” tương ứng với học sinh đó 6. Hệ thống sẽ chuyển sang trang chi tiết thông tin người dùng 7. Người quản trị nhập thông tin mới và nhấn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu lưu thành công học sinh. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Thông tin học sinh đã được cập nhật |
| Điểm mở rộng | Không có |

Bảng 3.12: Cập nhật thông tin người dung

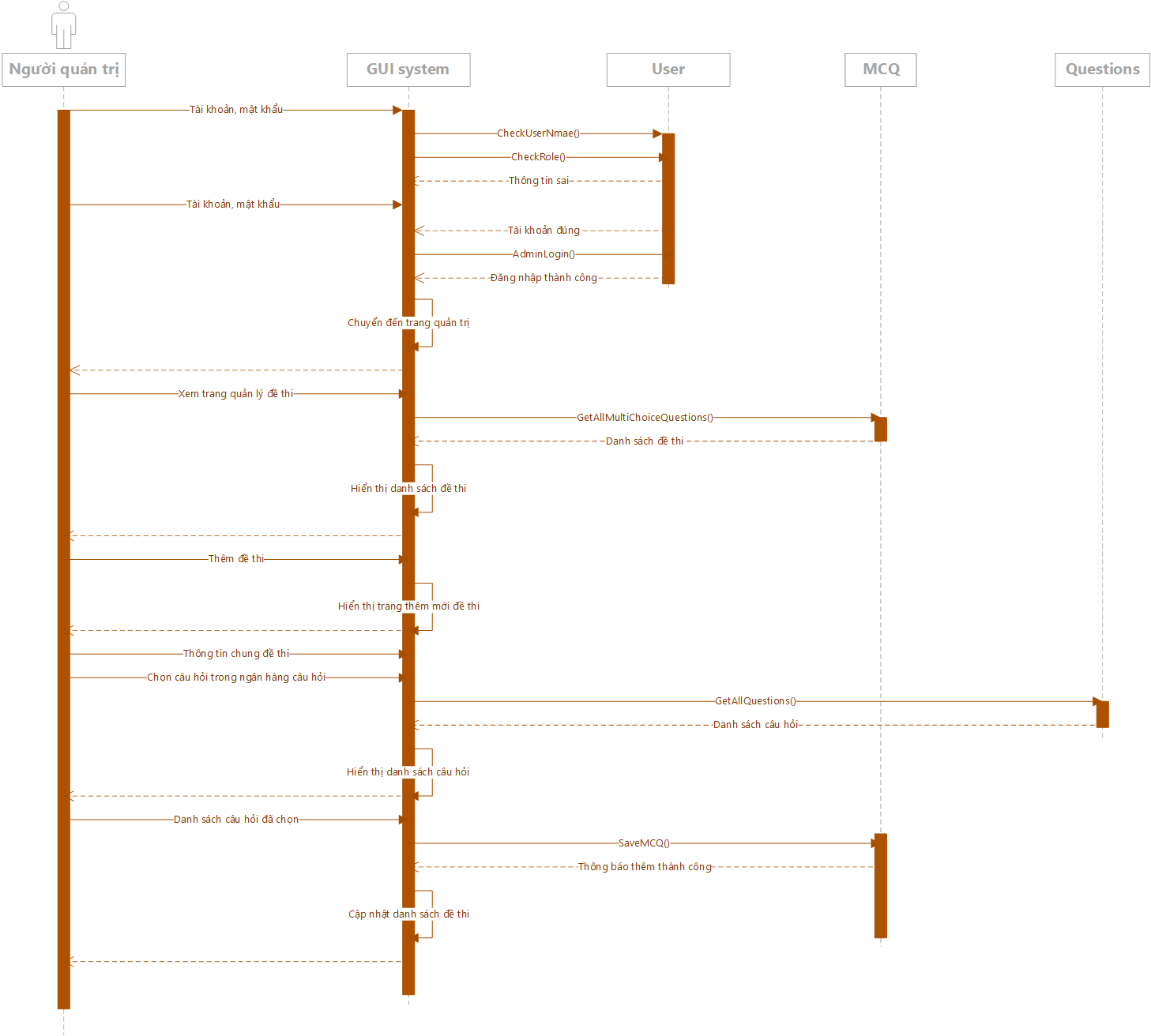


Hình 3.9: Cập nhật thông tin người dung

#### Thêm đề thi

|  |  |
| --- | --- |
| Tên chức năng | Thêm đề thi |
| Tóm tắt | Chức năng cho phép người quản trị thêm mới đề thi vào hệ thống |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý đề thi” 4. Hệ thống sẽ hiện thị danh sách đề thi hiện tại của hệ thống 5. Chọn nut “Thêm” 6. Hệ thống sẽ hiển thị trang thêm nội dung chi tiết của đề thi 7. Người quản trị nhập các thông tin chung của đề thi 8. Chọn “Thêm câu hỏi” 9. Hệ thống sẽ hiển thị danh sách các câu hỏi đã có để người quản trị chọn 10. Người quản trị đánh dấu các câu hỏi muốn thêm vào đề thi     1. Hoặc chọn nút “Thêm mới câu hỏi” để thêm câu hỏi vào ngân hàng câu hỏi và đánh dấu câu hỏi đó 11. Sau khi chọn các câu hỏi cần thêm vào dề thi , người quản trị chọn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu Lưu đề thi thành công. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Đề thi mới dược thêm vào hệ thống |
| Điểm mở rộng | Không có |

Bảng 3.13: Thêm đề thi



Hình 3.10: Sequence diagram thêm đề thi

#### Xóa đề thi

|  |  |
| --- | --- |
| Tên chức năng | Xóa đề thi |
| Tóm tắt | Chức năng cho phép người quản trị xóa đề thi khỏi hệ thống |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý đề thi” 4. Hệ thống sẽ hiện thị danh sách đề thi hiện tại của hệ thống 5. Xác đề thi cần xóa và chọn nút “Xóa” tương ứng với đề thi đó   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công đề thi. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Thông tin đề thi đã được xóa khỏi hệ thống |
| Điểm mở rộng | Không có |

Bảng 3.14: Xóa đề thi

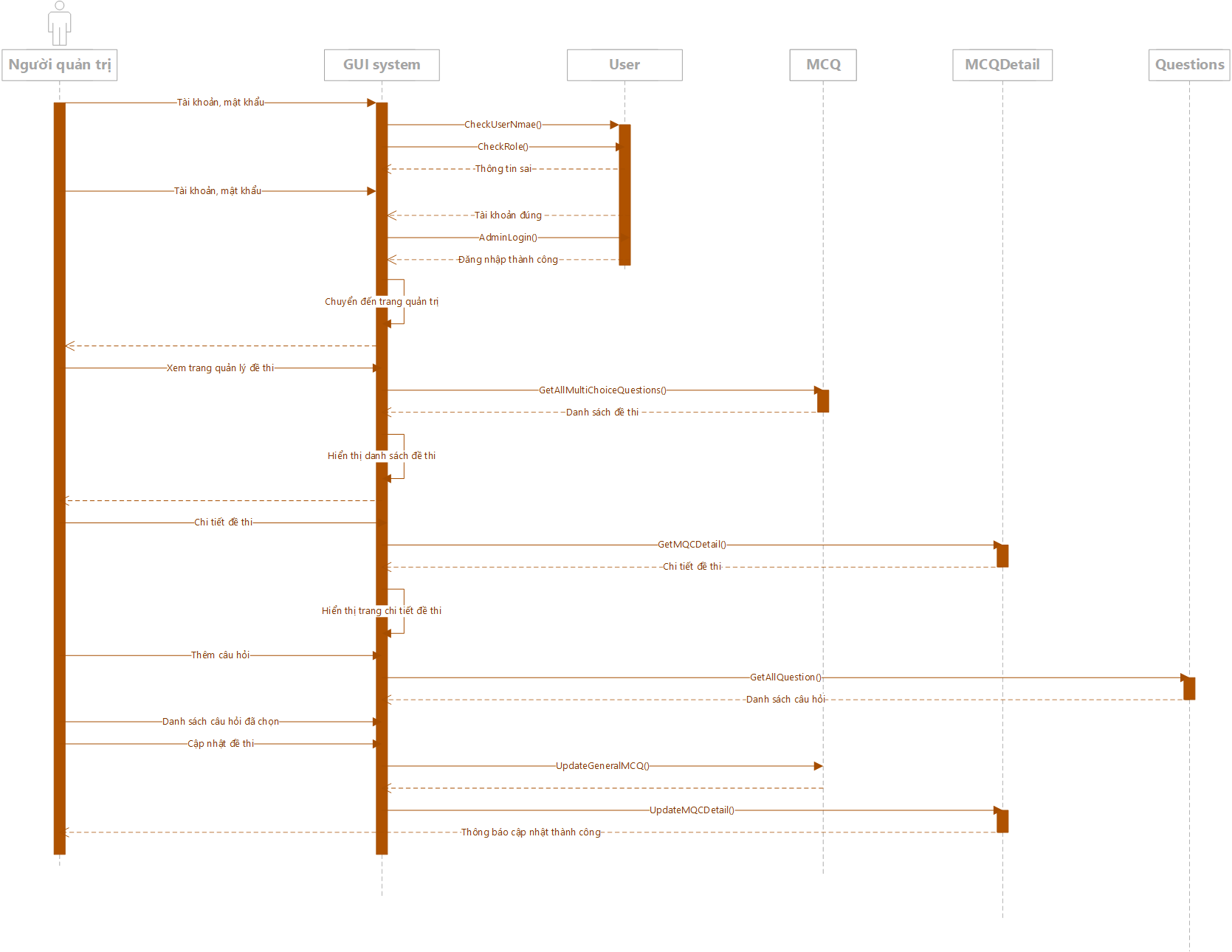
#### Cập nhật thông tin đề thi

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin chung của đề thi |
| Tóm tắt | Chức năng cho phép người quản trị cập nhật thông tin chung của đề thi |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý đề thi” 4. Hệ thống sẽ hiện thị danh sách đề thi hiện tại của hệ thống 5. Xác đề thi cần sửa và chọn nút “Sửa” tương ứng với đề thi đó 6. Hệ thống sẽ hiện trang chi tiết của đề thi ( thông tin chung và danh sách các câu hỏi và các đáp án ) 7. Người quản trị nhập thông tin mới cho đề thi và nhấn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu lưu thành công. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Đề thi được cập nhật thông tin |
| Điểm mở rộng | Không có |

Bảng 3.15: Cập nhật thông tin chung

|  |  |
| --- | --- |
| Tên chức năng | Thêm câu hỏi vào đề thi |
| Tóm tắt | Chức năng cho phép người quản trị thêm danh sách các câu hỏi vào đề thi đã có |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý đề thi” 4. Hệ thống sẽ hiện thị danh sách đề thi hiện tại của hệ thống 5. Xác đề thi cần sửa và chọn nút “Sửa” tương ứng với đề thi đó 6. Hệ thống sẽ hiện trang chi tiết của đề thi ( thông tin chung và danh sách các câu hỏi và các đáp án ) 7. Chọn “Thêm câu hỏi” 8. Hệ thống sẽ hiển thị danh sách các câu hỏi đã có để người quản trị chọn 9. Người quản trị đánh dấu các câu hỏi muốn thêm vào đề thi    1. Hoặc chọn nút “Thêm mới câu hỏi” để thêm câu hỏi vào ngân hàng câu hỏi và đánh dấu câu hỏi đó 10. Sau khi chọn các câu hỏi cần thêm vào dề thi , người quản trị chọn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu lưu thành công. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Đề thi đã thêm danh sách các câu hỏi |
| Điểm mở rộng | Không có |

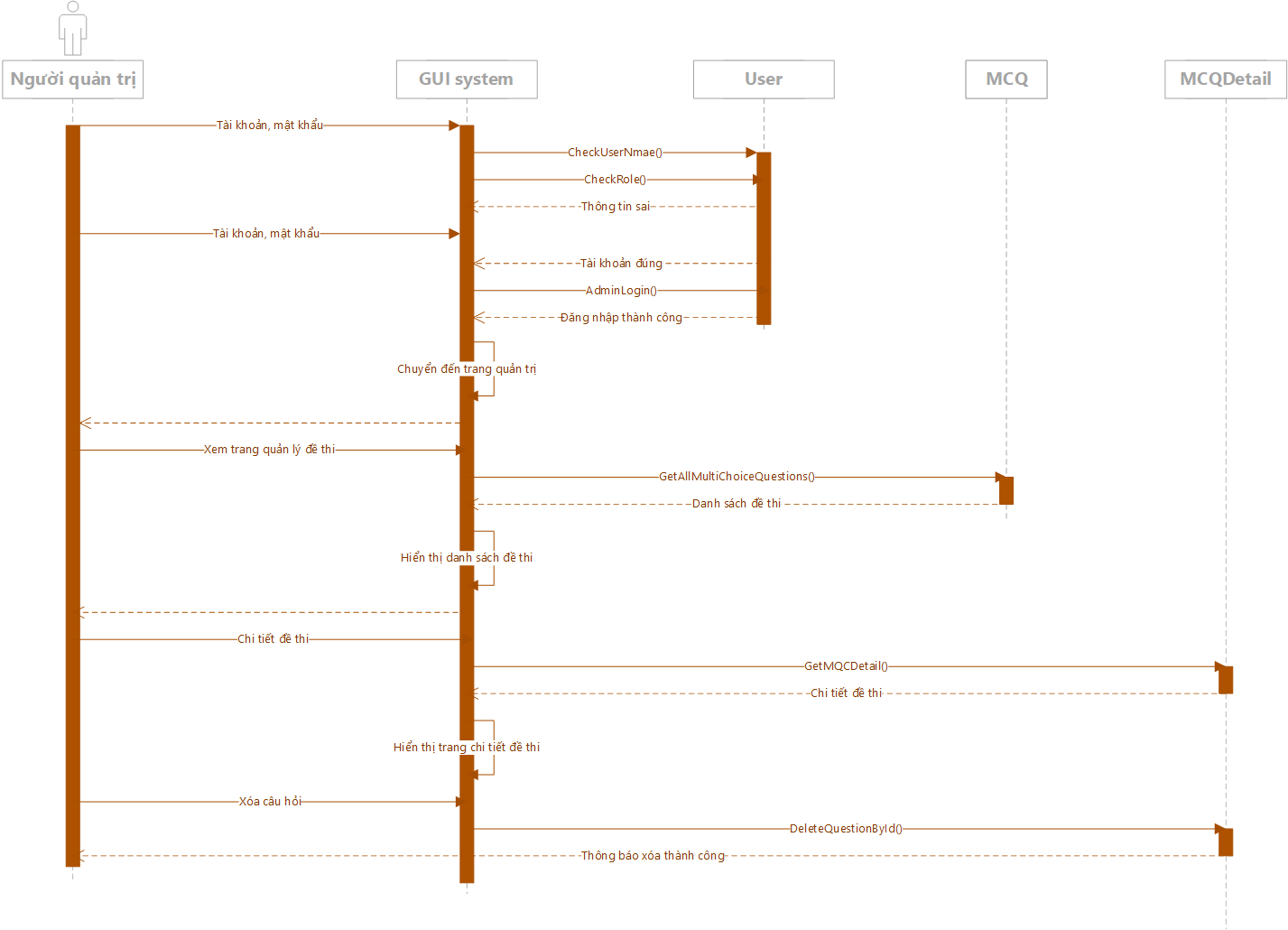
Bảng 3.16: Thêm câu hỏi vào đề thi



Hình 3.11: Sequence diagram thêm câu hỏi vào đề thi

|  |  |
| --- | --- |
| Tên chức năng | Xóa câu hỏi đề thi |
| Tóm tắt | Chức năng cho phép người quản trị xóa câu hỏi đề thi |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Quản lý đề thi” 4. Hệ thống sẽ hiện thị danh sách đề thi hiện tại của hệ thống 5. Xác đề thi cần sửa và chọn nút “Sửa” tương ứng với đề thi đó 6. Hệ thống sẽ hiện trang chi tiết của đề thi ( thông tin chung và danh sách các câu hỏi và các đáp án ) 7. Xác định câu hỏi cần xóa và chọn nút “Xóa” tương ứng với câu hỏi đó   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công câu hỏi của đề thi. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Câu hỏi bị xóa khỏi đề thi |
| Điểm mở rộng | Không có |

Bảng 3.17: Xóa câu hỏi của đề thi

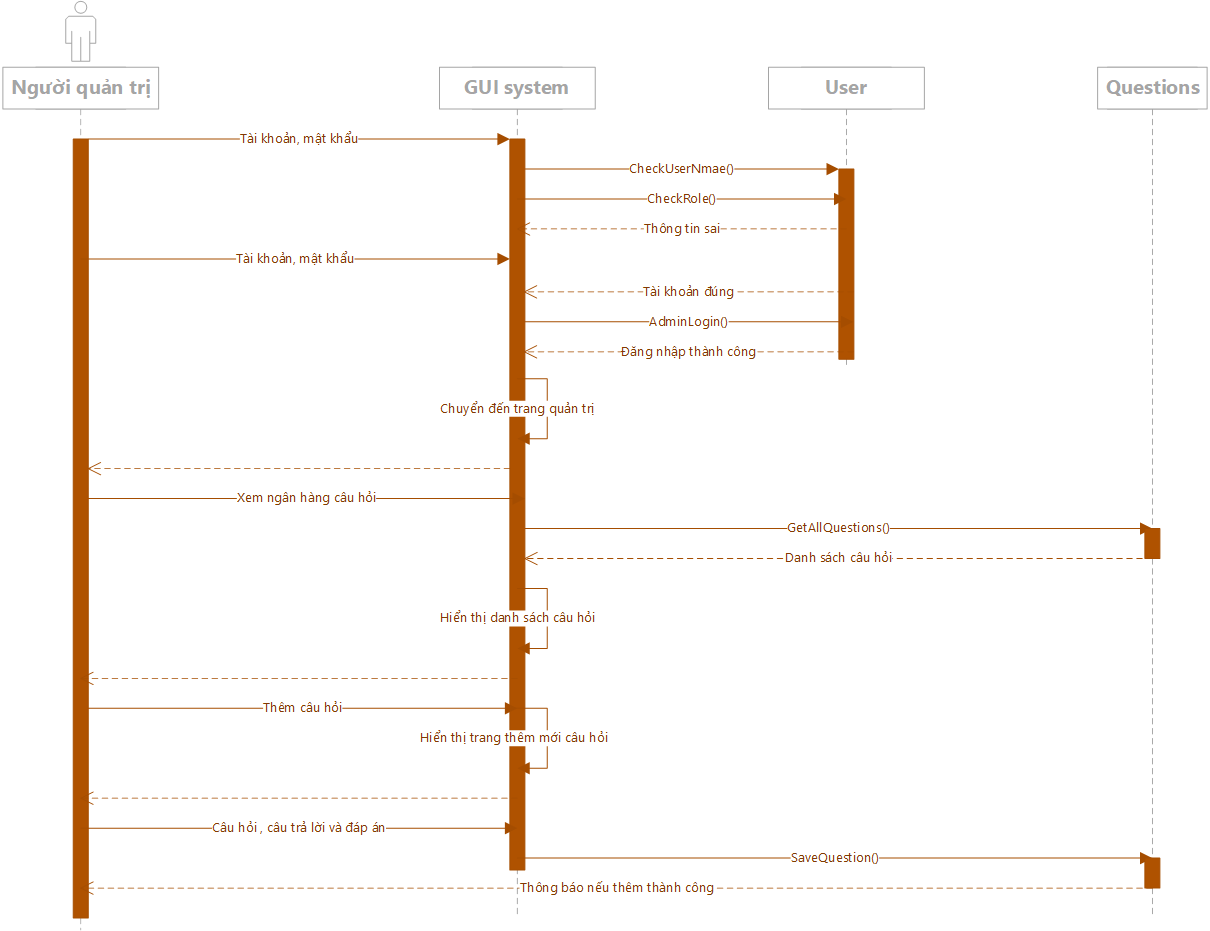


Hình 3.12: Sequence diagram xóa câu hỏi khỏi để thi

#### Thêm câu hỏi:

|  |  |
| --- | --- |
| Tên chức năng | Thêm câu hỏi |
| Tóm tắt | Chức năng cho phép người quản trị thêm mới câu hỏi vào ngân hàng câu hỏi |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Nhân hàng câu hỏi” 4. Hệ thống sẽ hiện thị danh sách câu hỏi hiện tại của hệ thống 5. Chọn nút “Thêm câu hỏi” 6. Hệ thống hiển thị trang thêm nội dung và câu trả lời cho câu hỏi 7. Người quản trị nhập nội dung và chọn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công học sinh. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Câu hỏi mới được thêm vào ngân hàng câu hỏi |
| Điểm mở rộng | Không có |

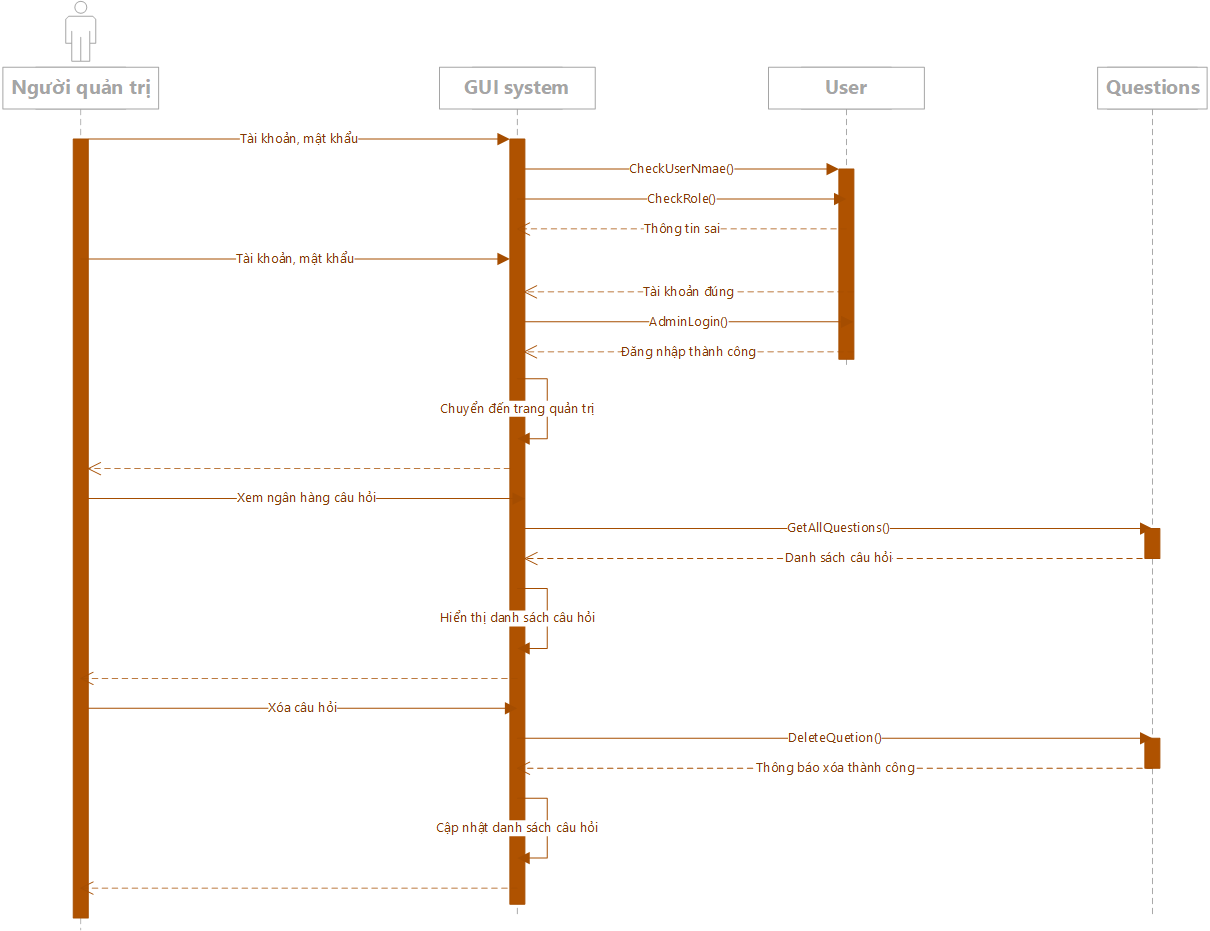
Bảng 3.18: Thêm câu hỏi



Hình 3.13: Sequence diagram thêm câu hỏi

|  |  |
| --- | --- |
| Tên chức năng | Thêm câu hỏi |
| Tóm tắt | Chức năng cho phép người quản trị xóa câu hỏi khỏi ngân hàng câu hỏi |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Nhân hàng câu hỏi” 4. Hệ thống sẽ hiện thị danh sách câu hỏi hiện tại của hệ thống 5. Người quản trị xác định câu hỏi cần xóa và chọn nút “Xóa” tương ứng với câu hỏi đó   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công học sinh. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Câu hỏi mới được thêm vào ngân hàng câu hỏi |
| Điểm mở rộng | Không có |

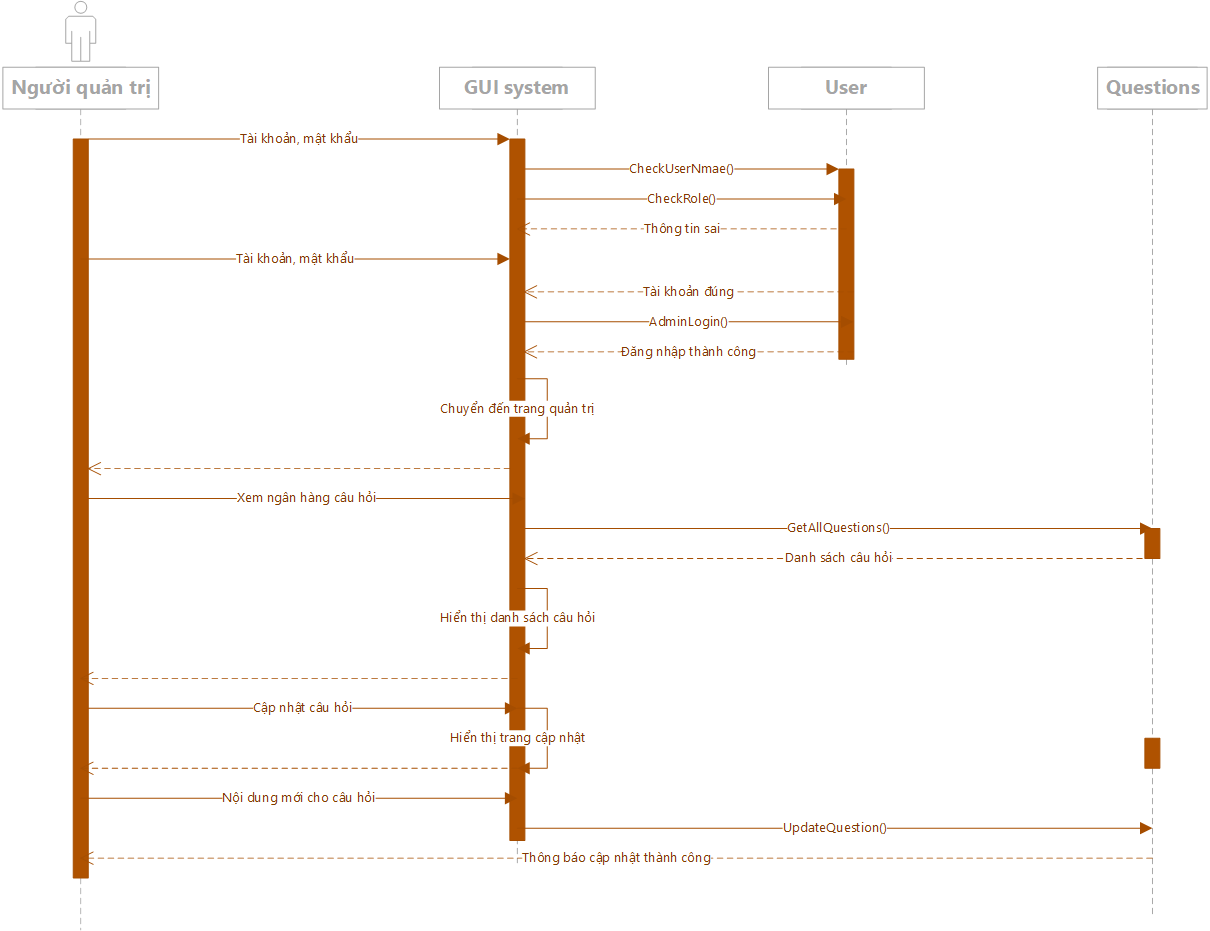
Bảng 3.19; Xóa câu hỏi



Hình 3.14: Sequence diagram xóa câu hỏi

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật nội dung và câu trả lời cho câu hỏi |
| Tóm tắt | Chức năng cho phép người quản trị Cập nhật nội dung và câu trả lời cho câu hỏi |
| Dòng sự kiện chính | 1. Người quản trị đăng nhập vào trang quản trị hệ thống ( dòng sự kiện khác : Đăng nhập không thành công ) 2. Hệ thống sẽ chuyển đến trang quản trị 3. Người quản trị chọn mục “Ngân hàng câu hỏi” 4. Hệ thống sẽ hiện thị danh sách câu hỏi hiện tại của hệ thống 5. Người quản trị xác định câu hỏi cần sửa và chọn nút “Sửa” tương ứng với câu hỏi đó 6. Hệ thống sẽ hiện trang chi tiết của câu hỏi gồm : câu hỏi , các câu đáp án , đáp án chính xác ) 7. Người quản trị cập nhật nội dung mới cho câu hỏi 8. Chọn nút “Lưu”   ( dòng dự kiện khác : thông báo ) |
| Dòng sự kiện khác | * Đăng nhập không thành công: Hiển thị thông báo lỗi nếu người dùng nhập sai thông tin tài khoản mật khẩu * Thông báo : thông báo thành công nếu xóa thành công học sinh. Hoặc thông báo lỗi nếu xảy ra lỗi trong quá trình xử lý |
| Các yêu cầu đặc biệt | Không có |
| Trạng thái hệ thống trước khi thực thiện use case | Actor: Người quản trị  Điều kiện: Đã đăng nhập thành công vào hệ thống |
| Trạng thái hệ thống sau khi thực hiện use case | Câu hỏi được cập nhật nội dung |
| Điểm mở rộng | Không có |

Bảng 3.20: Cập nhật nội dung và câu trả lời



Hình 3.15: Cập nhật thông tin câu hỏi

## Các công cụ và thư viện sử dụng:

* Nền tảng chính là ASP.NET Core
* Tích hợp fontend framework của javascript là Angular 5
* TypeScript dung như ngôn ngữ phát triển phía front-end , C# là ngôn ngữ phát triển phía back-end
* Sử dụng Entity framework core để truy xuất và thay đổi dữ liệu

## Kết chương:

Qua chương này, nhóm đã hiện thực được các chức năng của bài toán, cơ bản đã hoàn thành hai phần ba đoạn đường

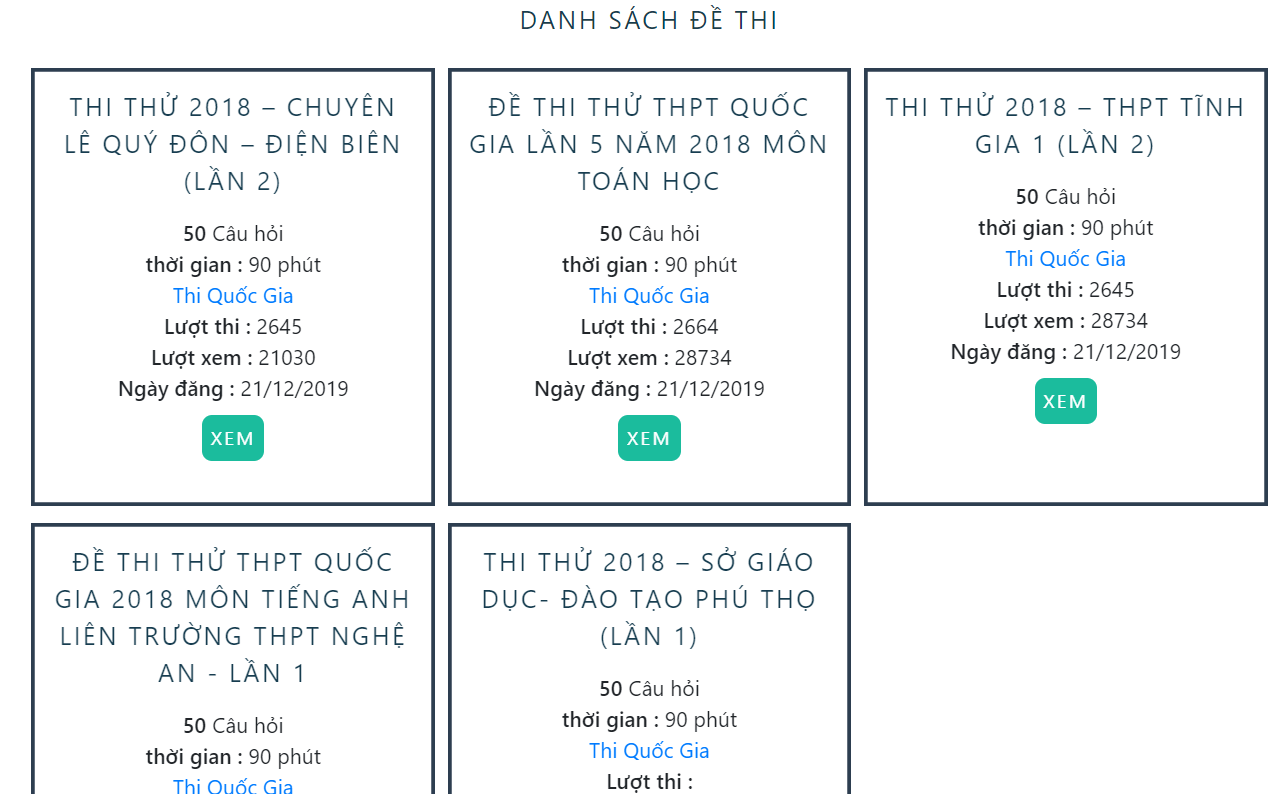
# Thử nghiệm và đánh giá

## Mở đầu:

Ở chương trước chúng ta đã phân tích qua các chức năng của hệ thống và hiện thực hóa các chức năng đó thông qua website. Chương này sẽ đi vào kiểm tra và đánh giá khả năng hoạt động của các chức năng ở trên.

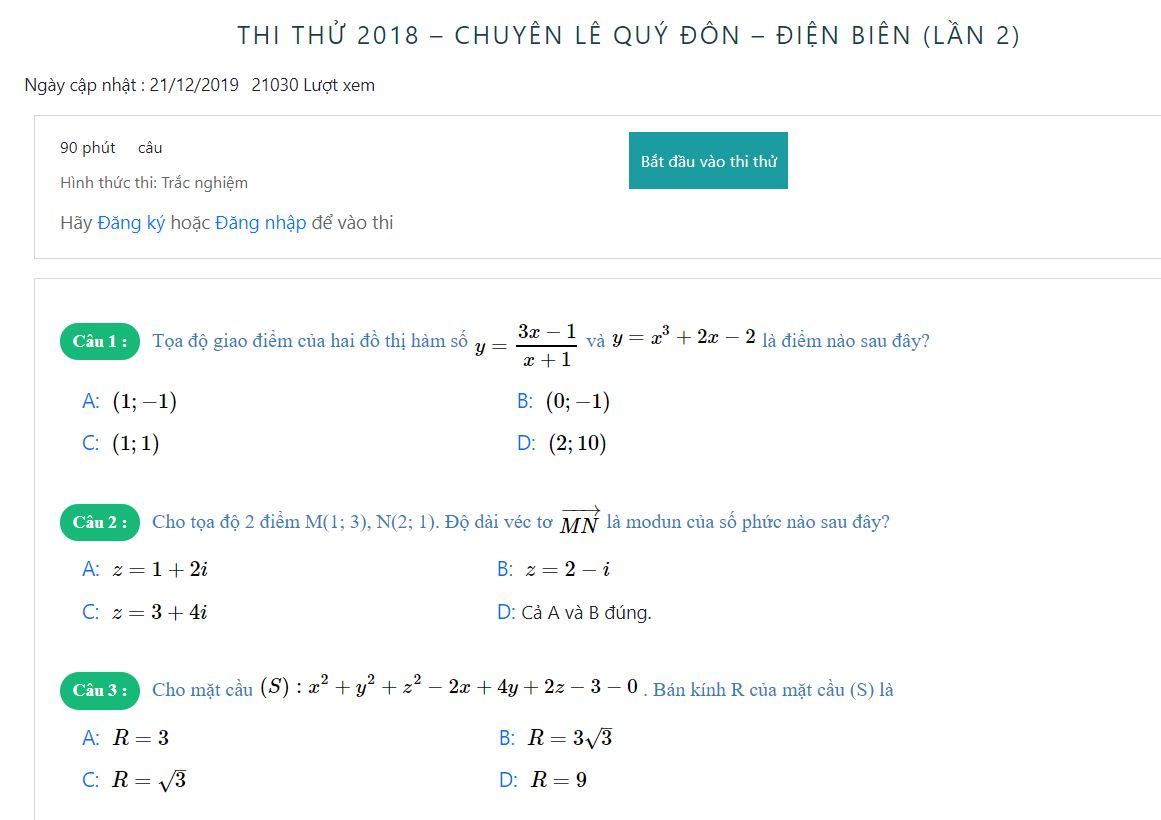
## Dữ liệu thử nghiệm:

* Thử nghiệm với bộ đề thi và bộ câu hỏi mẫu



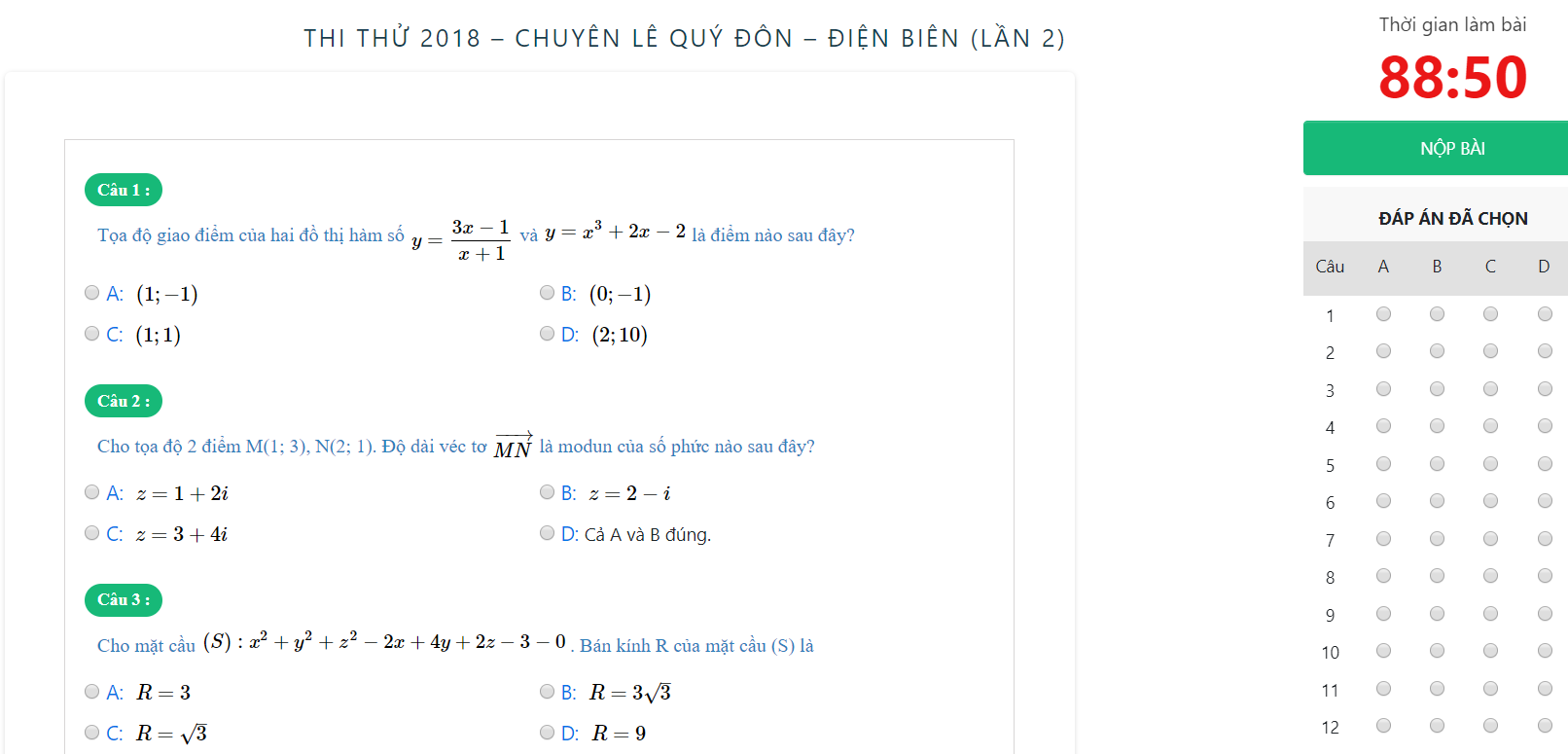
Hình 4.1: Danh sách đề thi

* Khi người dung vào trang chủ thì sẽ thấy danh sách các đề thi mới nhất. Nhấp chuột vào xem sẽ chuyển đến trang chi tiết của đề thi gôm thông tin đề thi , danh sách câu hỏi và câu trả lời

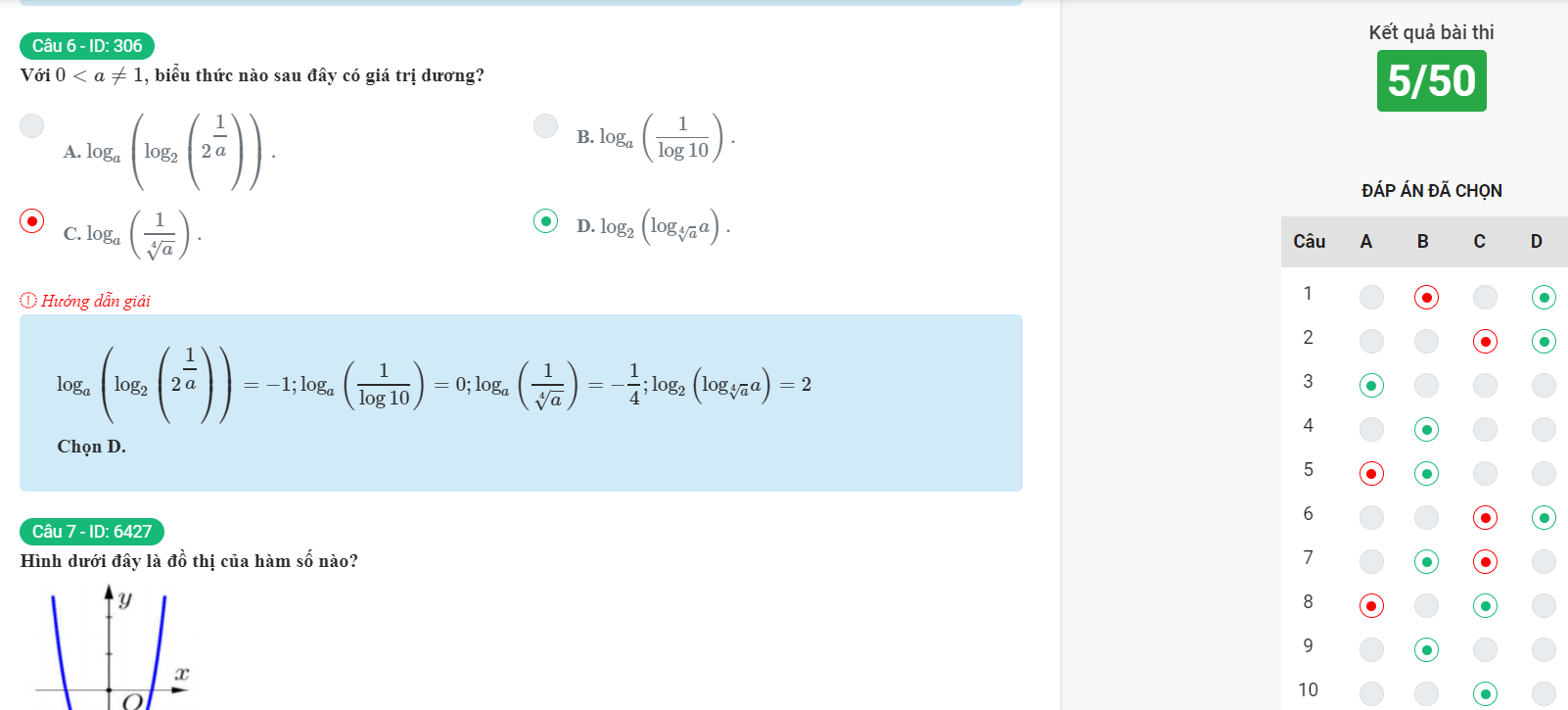


Hình 4.2: Đề thi mẫu

* Để có thể vào làm kiểm trả, người dung cần đăng ký tài khoản hoặc đăng nhập nếu đã có tài khoản
* Sau khi chọn thi thử thì website sẽ chuyển sang trang làm bài kiểm tra nếu người dung đã đăng nhập



Hình 4.3: Trang làm bài kiếm tra



Hình 4.4: Kết quả

## Kết quả:

Chúng ta có thể đánh giá thử nghiệm theo hai phương diện. Thức nhất là bộ dữ liệu thử nghiệm được lấy từ nhiều nguồn đa phần các chức năng đều hoạt động đúng yêu cầu. Thứ hai là việc làm kiếm tra thông qua các dữ liệu do người quản trị khởi tạo thông qua các chức năng dung cho trang quản trị. Với cách này chúng ta đã có thể kiểm tra được những trường hợp lỗi cũng như ngoại lệ để có được tính đúng đắn về chức năng của ứng dụng

## Kết chương:

Chương này đã trình bày dữ liệu thử nghiệm và kết quả thu được từ việc thử nghiệm. Qua đó, ta thấy các tính năng chính sử dụng cho học sinh đã được đáp ứng

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong luận văn này, qua việc nghiên cứu về một số kĩ thuật và công nghệ phát triển web và thực hiện xây dựng ứng dụng hỗ trợ thi toán trắc nghiệm thpt online dưới hình thức trắc nghiệm, nhóm đã có thêm nhiều kiến thức như:

* Có thêm kiến thức về mô hình MVC , diểu được vai trò và cách hoạt động và ứng dụng vào ASP.NET Core.
* Tìm hiểu về công nghệ phát triển web mới của Microsolf là ASP.NET Core. Áp dụng được các phương thức hỗ trợ của nó để xây dựng ứng dụng thực tế trên website
* Kiến thức cơ bản về Angular 5
* Ngôn ngữ TypeScript, một ngôn ngữ trung gian sẽ biên dịch ra JavaScript khi chạy nhưng ưu điểm là việc quản lý kiểu dữ liệu cũng như những lỗi cú pháp trong quá trình soạn thảo mã nguồn từ đó giảm thiểu lỗi xảy ra về sau.

Nhóm dự định trong thời gian tới sẽ tìm ra các giải pháp để tối ưu các thuật toán xử lý để nâng cao hiệu suất ứng dụng, kèm theo đó là khả năng tự động hóa các tính năng của trang quản trị để giúp người quản trị dễ dàng hơn trong việc quản lý và tạo đề thi mới… Nhóm cũng dẽ tiếp tục cải thiện UI/UX cho ứng dụng để người dùng có thể dễ dàng tương tác và có trải nghiệm tốt nhất. Để thực hiện việc này nhóm cần thêm thời gian cho việc tìm hiểu sâu hơn về các kĩ thuật phát triển và phương thức mà Angular và ASP.NET Core hỗ trợ đồng thời thực hiện khảo sát người dùng để có thể chỉnh sửa , bổ sung các tính năng nhằm đáp ứng nhu cầu của người dùng

**KẾT LUẬN**

Trong khóa luận này, em đã trình bày những hiểu biết về ASP.NET Core Framework, với trọng tâm là ASP.NET Core 2.2. Bên cạnh đó em cũng đã xây dựng một website dựa trên công nghệ ASP.NET Core.

Tài liệu này đưa ra những vấn đề chính sau:

* Giới thiệu về ASP.NET Core và làm quen với một ứng dụng của ASP.NET Core với mô hình MVC
* Tìm hiểu chi tiết về các thành phần chính Controller, View, Model binding, Model validation , middleware , cấu hình dự án
* Mô tả về chương trình ứng dụng website hỗ trợ cho việc thi toán trắc nghiệm thpt online

Khóa luận được thực hiện trong thời gian từ 14/10/2019 đến ngày 6/1/2020. Hy vọng với việc thực hiện đề tài này, em sẽ cung cấp một tài liệu tiếng Việt cơ bản và đầy đủ về công nghệ ASP.NET Core, giúp ích cho những người muốn tìm hiểu về nó sau này.

Một lần nữa, em xin chân thành cảm ơn quý thầy cô đã tạo điều kiện giúp đỡ em hoàn thành khóa luận này.

**Kết quả thu được**

* Năng cao hiểu biết về một công nghệ tiên tiến
* Cải thiện khả năng viết, dịch và tra cứu tài liệu
* Xây dựng một ứng dụng với những chức năng cần thiết

**Hướng phát triển**

* Tìm hiểu những kiến thức mới về ASP.NET Core
* Tìm hiểu những yêu cầu thực tế phát sinh đối với chương trình ứng dụng, qua đó cải thiện những chức năng sao cho tối ưu nhất có thể

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Tài liệu Introduction to ASP.NET Core của Microsoft |
| [2] | Báo cáo về ASP.NET MVC của Trần Quang Duy, 2013. |
| [3] | Khóa học về lập trình ASP.NET Core của TEDU :  <https://tedu.com.vn/khoa-hoc/khoa-hoc-aspnet-core-co-ban-14.html> |
| [4] | Angular 5 vs ASP.NET Core, Pablo Albella, 2018 |