**CHƯƠNG 1: TỔNG QUAN**

**1.1 Lý do chọn đề tài:**

**Xu hướng thị trường:** Ngành giáo dục Việt Nam đang ngày càng áp dụng các giải pháp kỹ thuật số, đặc biệt là trong và sau đại dịch COVID-19 đã chuyển đổi hình thức học tập truyền thống sang các nền tảng trực tuyến, trong đó hình thức thi trắc nghiệm xuất hiện nhiều hơn.

**Tính ứng dụng cao:** Hình thức thi trắc nghiệm ngày càng được áp dụng rộng rãi trong nhiều kỳ thi quan trọng, từ các bậc học phổ thông như kỳ thi THPT, tuyển sinh đại học, đến các chứng chỉ quốc tế như IELTS, TOEIC, SAT hay các chứng chỉ nghề nghiệp.

**Phương pháp đánh giá:** Thi bằng hình thức trắc nghiệm tránh các yếu tố thiên vị hoặc các đánh giá chủ quan của người chấm cũng như các đáp án được công khai một cách minh bạch và rõ ràng hơn so với hình thức tự luận.

**Mô hình kiến trúc:** Sử dụng kiến trúc Microservices giúp tối ưu hiệu suất, dễ dàng mở rộng hệ thống và cải thiện khả năng bảo trì. Mỗi dịch vụ hoạt động độc lập, giúp xử lý số lượng lớn người dùng đồng thời mà không ảnh hưởng đến toàn bộ hệ thống.

**1.2 Mục tiêu tổng quát:**  
Xây dựng website thi trắc nghiệm nhằm hỗ trợ người dùng trong việc đánh giá mức độ tiếp thu và khả năng hiểu biết của học sinh, sinh viên sau quá trình học tập, bao gồm các chức năng quản lí các thông tin sơ yếu lý lịch, ca thi và chi tiết đề thi.

**1.3 Mục tiêu cụ thể:**

**1.3.1 Framework (Blazor)**

Blazor là một framework phát triển ứng dụng web hiện đại sử dụng ngôn ngữ C#. Việc sử dụng Blazor mang lại nhiều lợi ích, bao gồm:

Hưởng lợi từ hệ sinh thái .NET:

Entity Framework Core: Hỗ trợ thao tác với cơ sở dữ liệu hiệu quả, cung cấp các phương thức truy vấn mạnh mẽ.

LINQ (Language Integrated Query): Cho phép truy vấn dữ liệu một cách trực quan và dễ dàng.

SignalR: Cung cấp khả năng giao tiếp real-time, giúp xây dựng các tính năng như thông báo tức thì hoặc chat trực tuyến.

Hỗ trợ kiến trúc Microservices.

Có thể tích hợp với RESTful API , gRPC để xây dựng hệ thống linh hoạt, dễ mở rộng và các services có thể tìm thấy và giao tiếp với nhau cũng như hỗ trợ kết nối giữa client và server.

**1.3.2 Frontend (Component Blazor – Razor Components)**

Blazor cung cấp mô hình phát triển Component-Based, giúp tạo ra các giao diện người dùng linh hoạt và dễ bảo trì:

Các trang web được chia thành nhiều thành phần nhỏ (component), có thể tái sử dụng trên nhiều trang khác nhau.

Các thành phần (component) được viết bằng Razor, kết hợp giữa HTML và C#, giúp dễ dàng quản lý và mở rộng.

Hỗ trợ Data Binding, Event Handling, giúp tối ưu hóa trải nghiệm người dùng.

**1.3.3 Backend (C#):**

Hệ thống backend được xây dựng trên nền tảng C# và ASP.NET Core, cung cấp các chức năng quản lý dữ liệu và xử lý nghiệp vụ cho hệ thống thi trắc nghiệm:

**Quản lý thông tin người dùng và tài khoản:**

Bảo mật tài khoản của người dùng bằng hàm băm Bcrypt và xác thực thông qua JWT Token.

Sinh viên không thể đăng nhập đồng thời cùng một tài khoản trên cả hai máy và quản trị viên có quyền đăng xuất tài khoản ra khỏi thiết bị của người dùng.

Đảm bảo mã xác thực chỉ lưu trữ tạm thời trong một tab (Session Storage) với thời gian tồn tại nhất định.

**Quản lí sinh viên:**

Quản trị viên có thể tra cứu thời gian thí sinh đăng nhập, và trạng thái thi bao gồm chưa làm hoặc bỏ thi, đang làm hoặc đã hoàn thành.

Cộng giờ khi thí sinh bị treo máy và reset đăng nhập khi tài khoản đã được đăng nhập trước đó hoặc có lý do chính đáng khác.

**Quản lí ca thi:**

Thống kê các ca thi đang được kích hoạt, thời gian làm bài của ca thi, số lượng của các thí sinh trong một ca và mã số đề của từng thí sinh đó.

Kích hoạt ca thi, đóng băng, hủy ca thi, kết thúc ca thi, từ quản trị viên.

Thêm thí sinh vào ca thi bằng file Excel hoặc nhập tay và xuất bảng điểm của các thí sinh trong một ca thi.

**Quản lý đề thi:**

Vì nội dung của đề thi không thay đổi theo thời gian, việc sử dụng Redis Cache giúp tối ưu hóa quá trình tải đề, giảm độ trễ và cải thiện hiệu suất hệ thống.

Cho phép cập nhật thông tin nội dung và thứ tự xuất hiện cho các nhóm câu hỏi, câu hỏi và câu trả lời.

Xáo trộn đề thi gốc thành các đề thi hoán vị theo ma trận đề thi dựa trên số câu lấy của từng chương/ phần và thứ tự giữa các câu có theo chương hay không.

**Quản lý bài thi:**

Hệ thống tự động lưu bài làm của thí sinh sau mỗi 2 phút. Do số lượng thí sinh đông, dữ liệu được đưa vào hàng đợi bằng RabbitMQ để phân luồng xử lý, giúp giảm tải và đảm bảo hiệu suất hệ thống.

Trong trường hợp treo máy hoặc gặp sự cố chính đáng, thí sinh có thể tiếp tục bài làm từ lần lưu dữ liệu gần nhất trong cơ sở dữ liệu, đảm bảo không mất tiến trình đã thực hiện.

**1.4 Tổng quan/cơ sở lý luận:**

Trong thời đại công nghệ số phát triển mạnh mẽ, nhu cầu ứng dụng công nghệ vào giáo dục ngày càng trở nên cấp thiết. Việc tổ chức các kỳ thi trực tuyến không chỉ giúp tiết kiệm thời gian, giảm thiểu chi phí mà còn mang lại nhiều lợi ích trong việc đánh giá chính xác năng lực của học viên.

Hệ thống thi trắc nghiệm trực tuyến là một giải pháp hữu hiệu, giúp học viên kiểm tra kiến thức, rèn luyện kỹ năng làm bài và cải thiện kết quả học tập. Các nghiên cứu đã chỉ ra rằng việc áp dụng các nền tảng học tập trực tuyến có thể nâng cao hiệu quả tiếp thu kiến thức.

Blazor, một framework hiện đại của Microsoft, kết hợp với ASP.NET Core, cung cấp một nền tảng mạnh mẽ để xây dựng hệ thống thi trắc nghiệm với hiệu suất cao và khả năng mở rộng linh hoạt. Hệ thống này không chỉ hỗ trợ người học mà còn giúp giáo viên dễ dàng quản lý, theo dõi và đánh giá năng lực của học viên một cách chính xác.

**CHƯƠNG 2. NỘI DUNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU:**

**2.1 Đối tượng nghiên cứu**

Hệ thống thi trắc nghiệm trực tuyến dành cho giáo viên, tổ chức giáo dục và học sinh, sinh viên. Giáo viên có thể quản lý đề thi, tổ chức thi và theo dõi kết quả. Học sinh, sinh viên có thể tham gia các kỳ thi trực tuyến, xem kết quả và đánh giá năng lực học tập của mình.

**2.2 Phạm vi nghiên cứu**

**2.2.1 Phạm vi chức năng:**

Xây dựng hệ thống thi trắc nghiệm trực tuyến dựa trên nền tảng Blazor WebAssembly và ASP.NET Core.

Triển khai các chức năng chính như quản lý kỳ thi, tạo đề thi tự động, chấm điểm trực tuyến và báo cáo kết quả.

Đánh giá hiệu suất hệ thống và khả năng mở rộng khi có số lượng người dùng lớn.

**2.2.2 Phạm vi công nghệ:**

Frontend: Blazor Component, HTML, CSS, JavaScript.

Backend: C#, RESTful API, JWT Token, Redis Cache, Rabbit MQ, SignalR.

Cơ sở dữ liệu: MSSQL.

**2.3. Nội dung nghiên cứu**

**2.3.1 Phân tích yêu cầu:**

Khảo sát nhu cầu người dùng thông qua bảng hỏi và phỏng vấn.

Nghiên cứu các website được sử dụng trước đó (ExamSuite, Tracnghiemonline).

Phân tích các vấn đề, giải quyết các rủi ro xảy ra khi xây dựng một hệ thống thi trắc nghiệm bằng việc nghiên cứu các thuật toán, công cụ, phần mềm hoặc công nghệ.

**2.3.2 Thiết kế hệ thống:**

Sơ đồ use-case, ERD, luồng xử lý thông tin ca thi, câu hỏi và câu trả lời.

Thiết kế UI/UX trên Figma.

**2.3.3 Phát triển ứng dụng:**

**Frontend**: Xây dựng giao diện người dùng với các màn hình chính như đăng nhập sinh viên, đăng nhập quản trị viên, hiển thị thông tin ca thi, thực hiện bài thi và quản lý ca thi.

**Backend**: Triển khai API phục vụ quản lý đề thi, truy xuất thông tin sinh viên, cập nhật trạng thái ca thi và các chức năng liên quan.

**Caching Database**: Tích hợp Redis Cache để tối ưu tốc độ truy xuất đề thi, giảm tải cho hệ thống cơ sở dữ liệu bằng cách hạn chế truy vấn lặp lại.

**Message Queue**: Sử dụng RabbitMQ để xử lý hàng đợi lưu bài thi của thí sinh, tránh tình trạng quá tải hệ thống khi số lượng người thi lớn.

**Cập nhật Real-time**: Áp dụng SignalR để đồng bộ dữ liệu theo thời gian thực giữa client và server, giúp cập nhật trạng thái bài thi, thời gian làm bài và các sự kiện quan trọng một cách mượt mà.

**Kiến trúc hệ thống**: Triển khai cấu trúc Microservices để xây dựng thành một tập hợp các dịch nhỏ, độc lập nhau trong việc xác thực tài khoản người dùng, lấy đề thi, cập nhật bài thi của thí sinh.

**2.4 Phương pháp nghiên cứu**

**Phương pháp nghiên cứu tài liệu**: Tìm hiểu các tài liệu liên quan đến công nghệ Blazor, ASP.NET Core và các hệ thống thi trực tuyến hiện có.

**Phương pháp phân tích, thiết kế hệ thống**: Sử dụng mô hình MVC (Model-View-Controller) kết hợp với kiến trúc Microservices để xây dựng hệ thống linh hoạt, dễ mở rộng.

**Phương pháp thử nghiệm và đánh giá**: Kiểm thử hệ thống bằng các kỹ thuật như Unit Testing, Load Testing để đảm bảo hiệu suất và tính ổn định của ứng dụng.

**Phương pháp so sánh**: Đối chiếu hiệu quả của hệ thống với các nền tảng thi trực tuyến khác để đánh giá ưu, nhược điểm của giải pháp.

**3.1. Dự kiến kết quả:**

**3.1.1 Sản phẩm ứng dụng:**

Học sinh, sinh viên có thể thi trên hệ thống và đưa ra kết quả chính xác với độ trễ thấp, tránh sự chờ đợi hoặc khựng thời gian lâu.

Việc quản trị viên có thể

Ứng dụng di động ORGAN MART trên App Store và Google Play.

Hệ thống quản trị web dành cho nhà cung cấp và admin.

**3.1.2 Tính năng nổi bật:**

Xem thông tin, nguồn gốc sản phẩm.

Cung cấp công thức nấu ăn cho người dùng.

Báo cáo doanh thu theo thời gian thực cho đối tác.

**CHƯƠNG 4: TÀI LIỆU THAM KHẢO**

Hương, T. T. (2022, 08 18). *Xu hướng hành vi mua của người tiêu dùng sau đại dịch Covid-19*. Retrieved from Tạp Chí Quản Lý Nhà Nước: https://www.quanlynhanuoc.vn/2022/08/18/xu-huong-hanh-vi-mua-cua-nguoi-tieu-dung-sau-dai-dich-covid-19

THỊNH, M. H. (2023, 04 12). *Xu hướng hành vi mua hàng trực tuyến của người tiêu dùng Việt Nam*. Retrieved from Tạp Chí Công Thương: https://tapchicongthuong.vn/xu-huong-hanh-vi-mua-hang-truc-tuyen-cua-nguoi-tieu-dung-viet-nam-104014.htm

File pdf nghiên cứu <https://medwinpublishers.com/PhIJ/which-one-do-you-choose-and-why-subjective-or-objective-examinations.pdf>

https://www.mdpi.com/2227-7390/9/23/3136

**7. Tiến độ thực hiện:**

**Tuần 1-2: Nghiên cứu & Triển khai dự án theo kiến trúc Microservices**

**Mục tiêu:** Xác định và tách nhỏ các dịch vụ trước đó thành các dịch vụ độc lập.

**Công việc:**

Nghiên cứu về kiến trúc Microservices và cách triển khai chúng.

Phân tách các dịch vụ thành các dịch vụ nhỏ đơn lẻ và độc lập với nhau.

Tìm hiểu và tích hợp Docker, Kubernetes vào dự án và luôn cập nhật và lưu trữ chúng trên Docker Hub.

Sử dụng các giao thức gRPC để giao tiếp giữa các services với nhau.

**Tuần 3-4: Tiếp tục hoàn thiện giao diện và chức năng của website**

**Mục tiêu:** Thống nhất lại các giao diện của các trang trước đó theo cùng một chủ đề và xây dựng trang quản lí đề thi.

**Công việc:**

Nghiên cứu và tìm hiểu về các giao diện và chức năng hiện đại bằng MudBlazor.

Thiết lập lại giao diện của các trang cũ đồng thời xây dựng các trang theo cùng một chủ đề.

Thiết kế UI/UX trang quản lí đề thi trên Figma.

Xây dựng trang hiển thị các nhóm câu hỏi, câu hỏi và câu trả lời theo từng đề thi mà người dùng đã chọn.

Thiết lập các thao tác thêm, xóa, sửa nội dung của đề thi theo nhu cầu của quản trị viên.

**Tuần 5-6: Hoàn thiện tất cả các tính năng & Báo cáo giữa kì.**

**Mục tiêu:** Thiết lập hoàn thiện các chức năng, giao diện, code và chuẩn bị các slide, tài liệu để báo cáo giữa kì.

**Công việc:**

Kiểm tra và tối ưu các dòng code đã làm, các giá trị, biến, logic hàm, thuật toán và khả năng bắt và xử lí lỗi.

Kiểm thử toàn bộ chức năng của hệ thống, cở sở dữ liệu, truy vấn, hàm.

Thiết kế và làm slide, chuẩn bị các câu hỏi có thể xảy ra trong buổi báo cáo.

Luyện tập và trau chuốt kĩ năng thuyết trình trước đám đông.

**Tuần 7-8: Xử lí trung gian Middleware và Tối ưu hóa API**

**Mục tiêu:** Xây dựng các giá trị trả về khi API bị lỗi và tối ưu hệ thống với RabbitMQ.

**Công việc:**

Nghiên cứu các lỗi có thể xảy ra khi các client giao tiếp với server và ngăn chặn việc gửi lỗi trả về API.

Tạo và viết mã hóa các lỗi thành các kí tự hoặc mã code khi trả về cho người dùng.

Thử nghiệm khả năng chịu tải của hệ thống với RabbitMQ từ đó đưa ra được số lần tối đa được phép yêu cầu cho server sau mỗi lần xử lý.

**Tuần 9-10: (Tuần dự phòng) Tiếp tục xử lí những nhiệm vụ chưa hoàn thành & Triển khai dự án lên Docker.**

**Mục tiêu:** Hoàn thiện tất cả các chức năng của hệ thống.

**Công việc:**

Kiểm tra và hoàn thiện các nhiệm vụ trước đó nếu chưa hoàn thành.

Triển khai dự án lên Docker và thử nghiệm chúng trên nhiều loại môi trường khác nhau.

**Tuần 11-12: Triển khai & Báo cáo tốt nghiệp**

**Mục tiêu:** Hoàn thiện sản phẩm và cuốn báo cáo.

**Công việc:**

Viết tài liệu hướng dẫn sử dụng và báo cáo kết quả.

Thiết kế và viết quyển báo cáo tốt nghiệp.

Chuẩn bị demo sản phẩm (video hoặc slide).

**Kết quả kỳ vọng sau 3 tháng**

**Sản phẩm khả dụng tối thiểu hoạt động được:**

Quản trị viên hoàn toàn kiểm soát được toàn bộ ca thi, thí sinh vào thi.

Quá trình thí sinh đăng nhập, vào thi và trả kết quả thi diễn ra với độ trễ thấp.

Kết quả điểm thí sinh nhận được là chính xác.

Đảm bảo các services không bị quá tải.

**Hướng phát triển tiếp theo:**

Xây dựng dự án quản lí ngân hàng câu hỏi để tạo đề thi.

Hỗ trợ nhiều ngôn ngữ và giao diện có chủ đề sáng và tối.

Triển khai dự án lên dịch vụ Azure App Service.