**TRƯỜNG ĐẠI HỌC VINH**

**VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**



**ĐỒ ÁN HỌC PHẦN  
PHÁT TRIỂN ỨNG DỤNG VỚI JAVA**

**XÂY DỰNG WEBSITE**

**VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH**

**THI TRẮC NGHIỆM KHÁCH QUAN**

**NHÓM: 13**

|  |  |
| --- | --- |
| GVHD: | ThS. Lê Quốc Anh |
| Nhóm sinh viên: | Đặng Ngọc Anh, 215748020110333 (NT) |
|  | Phạm Viết Minh, 215748020110140 |
|  |
|  |

**Nghệ An, 01/202****5**

MỞ ĐẦU

Với sự phát triển không ngừng của ngành công nghệ thông tin, trí tuệ nhân tạo hay cụ thể hơn là học máy thì các hệ thống quản lý thông minh đã và đang góp phần tối ưu hóa quy trình làm việc, nâng cao hiệu quả và đảm bảo tính minh bạch trong mọi hoạt động. Trong đó việc áp dụng các giải pháp công nghệ vào quản lý và vận hành quy trình của giáo dục ngày càng trở nên cần thiết. Hiểu rõ tầm quan trọng của việc ứng dụng kiến thức vào thực tiễn, chúng tôi đã lựa chọn triển khai thực hiện đồ án với đề tài "*Xây dựng Website vận hành và quản lý quy trình thi trắc nghiệm khách quan*" nhằm đưa ra một giải pháp hỗ trợ tốt hơn cho quá trình tổ chức, vận hành, quản lý và giám sát các kỳ thi trắc nghiệm trong môi trường giáo dục bậc đại học.

Qua đồ án này, chúng tôi tập trung xây dựng một hệ thống Website hướng đến việc tự động hoá toàn bộ quy trình tổ chức thi với các chức năng nổi bật như quản lý thông tin sinh viên, tổ chức thi trắc nghiệm, giám sát, và lưu trữ dữ liệu kết quả một cách tự động và hiệu quả. Đặc biệt, hệ thống sử dụng công nghệ thị giác máy tính để xác minh danh tính thí sinh, giám sát quá trình thi, nâng cao tính khách quan, tính chính xác và tính bảo mật thông tin.

Đồ án sử dụng ngôn ngữ lập trình Java với framework Spring Boot cho phần xử lý backend, framework Flask của Python để xử lý hình ảnh qua API, hệ quản trị cơ sở dữ liệu SQLite và cuối cùng là giao diện người dùng bằng HTML, CSS, và JavaScript. Việc tiếp cận với các framework và công nghệ đang được ứng dụng rộng rãi trong thực tế là bài học rất lớn không chỉ giúp ôn lại kiến thức đã học mà còn là cái nhìn tổng quát hơn cho định hướng lựa chọn công việc sau này.

Tuy nhiên chúng tôi nhận thức rõ rằng để xây dựng một Website hoàn chỉnh đòi hỏi đội ngũ lập trình viên có nhiều kinh nghiệm, sự nỗ lực, tính kỉ luật cao và khả năng giải quyết các thách thức phát sinh. Do đó sản phẩm trong phạm vi đồ án này không thể tránh khỏi nhiều thiếu sót, vì vậy nhóm mong nhận được sự đóng góp ý kiến của các thầy cô để có thể nhận ra, khắc phục và cải thiện hơn cho sản phẩm.

LỜI CẢM ƠN

Trước tiên, chúng tôi xin gửi lời cảm ơn sâu sắc và chân thành nhất đến ThS. Lê Quốc Anh, người đã tận tình hướng dẫn, đồng hành và hỗ trợ chúng tôi xuyên suốt quá trình thực hiện đồ án học phần Phát triển ứng dụng với Java. Trong suốt thời gian qua, thầy không chỉ truyền đạt những kiến thức nền tảng vững chắc mà còn chia sẻ nhiều chỉ dẫn thực tế quý báu, giúp chúng tôi có thể áp dụng lý thuyết vào thực tiễn một cách hiệu quả. Sự tận tâm và những góp ý chân thành của thầy đã giúp chúng tôi hoàn thành đồ án này, đồng thời mang lại cho chúng tôi nhiều bài học bổ ích và kinh nghiệm quý giá cho chặng đường nghề nghiệp sau này.

Cũng xin gửi lời cảm ơn đến toàn thể các thầy cô trong Viện Kỹ thuật Công nghệ, những người đã luôn tận tình giảng dạy, trang bị cho chúng tôi một nền tảng kiến thức vững chắc trong lĩnh vực công nghệ thông tin. Sự nhiệt huyết trong giảng dạy và tinh thần hỗ trợ của các thầy cô không chỉ giúp chúng tôi hiểu rõ các vấn đề học thuật mà còn khuyến khích chúng tôi sáng tạo và phát triển tư duy phản biện, tạo động lực mạnh mẽ để chúng tôi vững bước trên hành trình học tập và nghiên cứu.

Chúng tôi cũng không thể không nhắc đến sự đồng hành và nỗ lực của các thành viên trong nhóm, những người đã cùng nhau vượt qua những khó khăn và thử thách trong suốt quá trình làm việc. Đặc biệt, chúng tôi xin gửi lời cảm ơn sâu sắc đến bạn Phạm Viết Minh, người đã luôn sát cánh, hỗ trợ nhóm với tinh thần hợp tác, trách nhiệm và đoàn kết. Sự hỗ trợ lẫn nhau, sự kiên trì và nhiệt huyết của tất cả thành viên đã giúp chúng tôi hoàn thành đồ án này một cách tốt nhất.

Cuối cùng, chúng tôi chân thành mong muốn nhận được những góp ý, nhận xét từ quý thầy cô và các bạn để đồ án của chúng tôi ngày càng hoàn thiện và có giá trị thực tiễn hơn. Những ý kiến đóng góp ấy sẽ là nguồn động lực to lớn để chúng tôi tiếp tục phấn đấu và học hỏi trong tương lai.

Trân trọng cảm ơn,

*Đặng Ngọc Anh*,

*Phạm Viết Minh.*

DANH MỤC CÁC HÌNH

[Hình 1.1: Giao diện hệ thống kiểm tra thi trực tuyến 7](#_Toc187609135)

[Hình 1.2: Hình 1 ứng dụng xác minh khuôn mặt tại Vinpearl 8](#_Toc187609136)

[Hình 1.3: Hình 2 ứng dụng xác minh khuôn mặt tại Vinpearl 8](#_Toc187609137)

[Hình 2.1: Biểu đồ tuần tự chức năng cập nhật hình ảnh sinh viên 20](#_Toc187609138)

[Hình 2.2: Mẫu giao diện chức năng cập nhật hình ảnh sinh viên 21](#_Toc187609139)

[Hình 2.3: Biểu đồ ca sử dụng của giám thị trong chức năng so khớp thông tin 22](#_Toc187609140)

[Hình 2.4: Biểu đồ tuần tự chức năng đối chiếu thông tin với danh sách trong phòng thi](#_Toc187609141)

[Hình 2.5: Mẫu giao diện chức năng đối chiếu với các sinh viên trong phòng thi 26](#_Toc187609142)

[Hình 2.6: Biểu đồ ca sử dụng đối chiếu khuôn mặt của giám thị 27](#_Toc187609143)

[Hình 2.7: Biểu đồ ca sử dụng đối chiếu khuôn mặt của sinh viên 27](#_Toc187609144)

[Hình 2.8: Biểu đồ tuần tự chức năng đối chiếu khuôn mặt thực tế với ảnh thẻ sinh viên](#_Toc187609145)

[Hình 2.9: Mẫu giao diện chức năng đối chiếu khuôn mặt 31](#_Toc187609146)

[Hình 2.10: Biểu đồ tuần tự xuất phiếu thi và giám sát quy trình thi 34](#_Toc187609147)

[Hình 2.10: Biểu đồ ca sử dụng chức năng quản lý sinh viên 35](#_Toc187609148)

[Hình 2.11: Biều đồ tuần tự của chức năng quản lý sinh viên 37](#_Toc187609149)

[Hình 2.12: Mẫu giao diện trang quản lý sinh viên 38](#_Toc187609150)

[Hình 2.13: Biểu đồ ca sử dụng chức năng quản lý giám thị 39](#_Toc187609151)

[Hinh 2.14: Biểu đồ tuần tự của chức năng quản lý giám thị 42](#_Toc187609152)

[Hình 2.15: Mẫu giao diện chức năng quản lý giám thị 43](#_Toc187609153)

[Hình 2.16: Mẫu giao diện chức năng thêm mới giám thị 43](#_Toc187609154)

[Hình 2.17: Biểu đồ ca sử dụng chức năng quản lý phòng thi 44](#_Toc187609155)

[Hình 2.18: Mẫu giao diện chức năng quản lý phòng thi 46](#_Toc187609156)

[Hình 2.18: Biểu đồ ca sử dụng hệ thống 47](#_Toc187609157)

[Hình 3.1: Ảnh mã nguồn Student Entity 54](#_Toc187609158)

[Hình 3.2: Ảnh mã nguồn Student Repository 54](#_Toc187609159)

[Hình 3.3: Ảnh mã nguồn Student Service 56](#_Toc187609160)

[Hình 3.4: Ảnh mã nguồn Student Controller 58](#_Toc187609161)

[Hình 3.5: Giao diện chức năng cập nhật thông tin sinh viên 59](#_Toc187609162)

[Hình 3.6: Ảnh mã nguồn Mô-đun đọc danh sách từ cơ sở dữ liệu 64](#_Toc187609163)

[Hình 3.7: Giao diện chức năng so khớp thông tin sinh viên 65](#_Toc187609164)

[Hình 3.8: Ảnh mã nguồn Mô-đun nhận diện và cắt khuôn mặt có trên thẻ sinh viên 69](#_Toc187609165)

[Hình 3.9: Ảnh mã nguồn Mô-đun đối chiếu ảnh với khuôn mặt thực tế 71](#_Toc187609166)

[Hình 3.10: Giao diện chức năng nhận diện khuôn mặt 72](#_Toc187609167)

[Hình 3.11: Ảnh mã nguồn Mô-đun in phiếu thi cho sinh viên 76](#_Toc187609168)

[Hình 3.12: Giao diện trang chủ trang quản trị viên 77](#_Toc187609169)

[Hình 3.13: Giao diện trang quản lý sinh viên 78](#_Toc187609170)

[Hình 3.14: Ảnh mã nguồn Proctor Entity 80](#_Toc187609171)

[Hình 3.15: Ảnh mã nguồn Proctor Repository 81](#_Toc187609173)

[Hình 3.16: Ảnh mã nguồn Proctor Service 82](#_Toc187609175)

[Hình 3.17: Ảnh mã nguồn Proctor Controller 85](#_Toc187609177)

[Hình 3.18: Giao diện trang quản lý giám thị 85](#_Toc187609178)

[Hình 3.19: Ảnh mã nguồn ExamRoom Entity 87](#_Toc187609179)

[Hình 3.20: Ảnh mã nguồn ExamRoom Repository 87](#_Toc187609180)

[Hình 3.21: Ảnh mã nguồn ExamRoom Service 88](#_Toc187609181)

[Hình 3.22: Ảnh mã nguồn ExamRoom Controller 89](#_Toc187609182)

[Hình 3.23: Giao diện trang quản lý phòng thi 90](#_Toc187609183)

[Hình 3.24: Ảnh mã nguồn Course Entity 91](#_Toc187609184)

[Hình 3.25: Ảnh mã nguồn Course Repository 92](#_Toc187609185)

[Hình 3.26: Ảnh mã nguồn Course Service 92](#_Toc187609186)

[Hình 3.27: Ảnh mã nguồn Course Service 94](#_Toc187609187)

[Hình 3.28: Giao diện trang quản lý học phần 94](#_Toc187609188)

[Hình 3.29: Ảnh mã nguồn ExamSession Entity 95](#_Toc187609189)

[Hình 3.30: Ảnh mã nguồn ExamSession Repository 96](#_Toc187609190)

[Hình 3.31: Ảnh mã nguồn ExamSession Service 98](#_Toc187609191)

[Hình 3.32: Ảnh mã nguồn ExamSession Controller 100](#_Toc187609192)

[Hình 3.33: Giao diện trang quản lý phòng thi 100](#_Toc187609193)

DANH MỤC CÁC BẢNG

[Bảng 1.3: Bảng mô tả yêu cầu chức năng hệ thống 11](#_Toc187609227)

[Bảng 1.4: Bảng mô tả yêu cầu phi chức năng hệ thống 12](#_Toc187609228)

[Bảng 2.1: Bảng tóm tắt Sprint và mục tiêu hướng đến 14](#_Toc187609229)

[Bảng 2.2: Bảng mô tả dữ liệu lưu trữ trong tbl sinh viên 18](#_Toc187609230)

[Bảng 2.3: Bảng chức năng quản lý thông tin sinh viên 19](#_Toc187609231)

[Bảng 2.4: Bảng mô tả dữ liệu lưu trữ trong tbl ExamSessions 24](#_Toc187609232)

[Bảng 2.5: Bảng chức năng đối chiếu thông tin sinh viên với danh sách phòng thi 25](#_Toc187609233)

[Bảng 2.6: Bảng chức năng đối chiếu khuôn mặt thực tế với ảnh thẻ sinh viên. 29](#_Toc187609234)

[Bảng 2.7: Bảng chức năng xuất phiếu thi cá nhân cho sinh viên 33](#_Toc187609235)

[Bảng 2.8: Quy trình quản lý sinh viên của quản trị viên 37](#_Toc187609236)

[Bảng 2.9: Bảng mô tả dữ liệu lưu trữ trong tbl Proctor 40](#_Toc187609237)

[Bảng 2.10: Quy trình quản lý giám thị của quản trị viên 41](#_Toc187609238)

[Bảng 2.11: Bảng mô tả dữ liệu lưu trữ trong tbl ExamRooms 45](#_Toc187609239)

[Bảng 2.12: Quy trình quản lý phòng thi của quản trị viên 45](#_Toc187609240)

[Bảng 3.1: Kiểm thử chức năng xử lý ảnh đầu vào và trích xuất thông tin 61](#_Toc187609241)

[Bảng 3.2: Bảng mô tả Mô-đun lấy danh sách từ cơ sở dữ liệu 62](#_Toc187609242)

[Bảng 3.3: Kiểm thử chức năng so khớp thông tin sinh viên 67](#_Toc187609243)

[Bảng 3.3: Bảng mô tả Mô-đun nhận diện và cắt khuôn mặt có trên thẻ sinh viên 68](#_Toc187609244)

[Bảng 3.4: Bảng mô tả Mô-đun đối chiếu khuôn mặt đã cắt với khuôn mặt thực tế 71](#_Toc187609245)

[Bảng 3.4: Kiểm thử chức năng đối chiếu khuôn mặt với ảnh thực tế 74](#_Toc187609246)

[Bảng 3.5: Bảng mô tả Mô-đun in phiếu thi cho sinh viên 75](#_Toc187609247)

MỤC LỤC

[MỞ ĐẦU ii](#_Toc187609105)

[LỜI CẢM ƠN iii](#_Toc187609106)

[DANH MỤC CÁC HÌNH iv](#_Toc187609107)

[DANH MỤC CÁC BẢNG vi](#_Toc187609108)

[MỤC LỤC vii](#_Toc187609109)

[CHƯƠNG 1. MÔ TẢ BÀI TOÁN VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN 1](#_Toc187609110)

[1.1. Mô tả tổng quan bài toán 1](#_Toc187609111)

[1.2. Tìm hiểu bài toán vận hành và quản lý thi trắc nghiệm khách quan 2](#_Toc187609112)

[1.3. Một số ứng dụng liên quan đến đề tài 6](#_Toc187609113)

[1.4. Các yêu cầu cho hệ thống mới 9](#_Toc187609114)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN 13](#_Toc187609115)

[2.1. Phân tích yêu cầu và xác định tác nhân 13](#_Toc187609116)

[2.2. Sprint 1: Quản lý thông tin cá nhân sinh viên 16](#_Toc187609117)

[2.3. Sprint 2: Chức năng so khớp thông tin sinh viên với danh sách trong phòng thi 21](#_Toc187609118)

[2.4. Sprint 3: Đối chiếu khuôn mặt sinh viên trong cơ sở dữ liệu với ảnh thực tế 27](#_Toc187609119)

[2.5. Sprint 4: Xuất phiếu thi và giám sát quy trình thi 32](#_Toc187609120)

[2.6. Sprint 5: Các chức năng quản lý của quản trị viên 35](#_Toc187609121)

[2.7. Biểu đồ ca sử dụng 46](#_Toc187609122)

[CHƯƠNG 3. XÂY DỰNG HỆ THỐNG VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN 48](#_Toc187609123)

[3.1. Môi trường phát triển 48](#_Toc187609124)

[3.2. Chức năng cập nhật ảnh cá nhân của sinh viên lên hệ thống 52](#_Toc187609125)

[3.3. Xây dựng các thành phần liên quan 52](#_Toc187609126)

[3.4. Chức năng so khớp thông tin sinh viên với danh sách phòng thi 61](#_Toc187609127)

[3.5. Chức năng đối chiếu khuôn mặt trên thẻ sinh viên với ảnh thực tế 67](#_Toc187609128)

[3.6. Chức năng xuất phiếu và giám sát quy trình thi 74](#_Toc187609129)

[3.7. Các chức năng quản lý của quản trị viên 76](#_Toc187609130)

[KẾT LUẬN 101](#_Toc187609131)

[Kết quả đạt được và hướng phát triển 101](#_Toc187609132)

[Những tồn tại và hạn chế 101](#_Toc187609133)

[TÀI LIỆU THAM KHẢO 102](#_Toc187609134)

1. MÔ TẢ BÀI TOÁN VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN

Việc sử dụng hình thức thi trắc nghiệm khách quan để kiểm tra và đánh giá kết quả học tập của sinh viên đang được các cơ sở giáo dục bậc đại học lựa chọn, dù vậy vẫn còn tồn tại nhiều khó khăn và thách thức. Trong chương đầu tiên này, chúng tôi sẽ trình bày bối cảnh thực tế dẫn đến việc chọn lựa đề tài, nhấn mạnh những yêu cầu cần thiết để xây dựng một hệ thống quản lý thi hiệu quả, và phân tích những thách thức đặt ra khi triển khai. Qua đó, chương cũng sẽ nêu ra và giới thiệu sơ bộ các công nghệ được sử dụng để cải thiện và khắc phục các nhược điểm hiện tại.

* 1. Mô tả tổng quan bài toán

Thực tế, tổ chức thi trắc nghiệm khách quan cho số lượng lớn sinh viên đã mang lại nhiều cải tiến so với hình thức thi truyền thống, ưu điểm như chuẩn hóa cách đánh giá và giảm bớt áp lực chấm thi. Tuy nhiên, phương thức này vẫn còn nhiều điểm hạn chế cần khắc phục. Một trong những bất cập lớn nhất hiện nay là **quá trình xác minh danh tính sinh viên** trước mỗi ca thi vẫn chủ yếu được thực hiện thủ công. Phương pháp này không chỉ tiêu tốn thời gian mà còn dễ dẫn đến sai sót, nhất là với các kỳ thi có số lượng thí sinh tham gia lớn. Thêm vào đó, việc **phát phiếu thi chứa thông tin quan trọng** như tài khoản và mật khẩu truy cập thường diễn ra trước ca thi và trước cả bước xác minh danh tính. Đây là một lỗ hổng lớn trong khâu quản lý, tiềm ẩn nguy cơ gian lận, khi thông tin này có thể bị lạm dụng hoặc trao đổi trước giờ thi. Ở khâu **giám sát thi cử**, hầu hết các cơ sở giáo dục vẫn sử dụng phương pháp truyền thống, bố trí từ 1-2 giám thị cho mỗi phòng thi để quan sát trực tiếp sinh viên làm bài. Mặc dù phương pháp này đã được áp dụng từ lâu, nhưng nó đang bộc lộ nhiều hạn chế trước sự phát triển ngày càng tinh vi của các hành vi gian lận. Những bất cập trong quá trình vận hành này không chỉ làm giảm hiệu quả mà còn gây lãng phí thời gian, nguồn lực và thậm chí có thể ảnh hưởng đến uy tín của tổ chức giáo dục. Từ thực trạng trên việc ứng dụng các **giải pháp công nghệ hiện đại** để tự động hóa quy trình thi là một đòi hỏi cấp thiết. Các công nghệ như nhận diện khuôn mặt để xác minh danh tính, hệ thống giám sát thông minh sử dụng trí tuệ nhân tạo, hoặc phần mềm bảo mật để quản lý dữ liệu thi đều có thể giúp khắc phục những hạn chế hiện tại. Những giải pháp này không chỉ giúp giảm thiểu tối đa sai sót và gian lận mà còn tối ưu hóa thời gian, nhân lực trong tổ chức thi. Quan trọng hơn, chúng tạo ra một môi trường thi cử minh bạch, công bằng và hiệu quả hơn, góp phần nâng cao chất lượng giáo dục cũng như sự tin cậy của xã hội đối với hệ thống giáo dục hiện đại.

* 1. Tìm hiểu bài toán vận hành và quản lý thi trắc nghiệm khách quan

Hiện tại, trong công cuộc tổ chức thi trắc nghiệm khách quan yêu cầu giám thị thực hiện tuần tự nhiều bước nhằm đảm bảo tính chính xác và minh bạch. Đầu tiên, giám thị nhận danh sách sinh viên thông qua văn bản giấy được cấp phát ngay trước giờ thi, sau đó gọi tên từng sinh viên để xác nhận sự có mặt. Tiếp theo, giám thị đối chiếu thông tin cá nhân trên thẻ sinh viên, bao gồm họ tên và mã số, với danh sách để đảm bảo khớp đúng. Đồng thời, giám thị kiểm tra khuôn mặt thực tế của sinh viên với ảnh trên thẻ nhằm xác minh danh tính. Chỉ khi hoàn thành đầy đủ các bước xác minh này, sinh viên mới nhận được phiếu thi để tham gia thực hiện bài thi, nhằm đảm bảo quá trình diễn ra chặt chẽ, hạn chế gian lận và sai sót.

Qua thực tiễn từ bài toán này, để hướng đến tự động hóa và tối ưu quy trình, nhóm chúng tôi đề xuất các bước thực hiện theo thứ tự ưu tiên dựa trên mức độ quan trọng, tạo cơ sở cho các bước tiếp theo:

* + 1. Mô tả thông tin sinh viên

Trong thực tế, khi chưa có hệ thống quản lý thông tin sinh viên, việc thu thập và lưu trữ dữ liệu thường được thực hiện thủ công thông qua hồ sơ giấy hoặc bảng tính Excel. Các thông tin như họ tên, mã sinh viên, ngày sinh, giới tính, số điện thoại, email, khoa/viện, lớp học và khóa học được ghi chép rời rạc, dễ bị thất lạc hoặc nhầm lẫn. Điều này gây khó khăn cho công tác quản lý hành chính, đặc biệt khi cần sắp xếp lịch học, lịch thi hoặc tổ chức các hoạt động ngoại khóa.

Việc cập nhật trạng thái học tập của sinh viên như đang học, đã tốt nghiệp hay tạm ngưng học cũng gặp nhiều bất tiện. Nhà trường phải kiểm tra và điều chỉnh thông tin trên nhiều tài liệu khác nhau, mất thời gian và dễ sai sót. Đặc biệt, trong các kỳ thi, việc điểm danh và xác minh danh tính sinh viên thường được thực hiện bằng cách đối chiếu giấy tờ tùy thân, dễ xảy ra gian lận và không đảm bảo tính công bằng.

Thiếu hệ thống quản lý tập trung cũng khiến việc lưu trữ và sử dụng ảnh thẻ sinh viên hay ảnh khuôn mặt phục vụ cho công tác xác minh danh tính gần như không khả thi. Điều này làm giảm hiệu quả giám sát và gây khó khăn trong việc tổ chức thi cử minh bạch, nghiêm túc.

* + 1. Mô tả thông tin giám thị

Khi chưa có hệ thống quản lý thông tin giám thị, công tác quản lý thường được thực hiện thủ công, gây ra nhiều bất tiện và hạn chế. Thông tin của giám thị như họ tên, đơn vị công tác, số điện thoại và email thường được lưu trữ dưới dạng văn bản giấy hoặc file Excel. Việc thu thập và cập nhật thông tin này mất nhiều thời gian và dễ xảy ra sai sót, đặc biệt khi cần chỉnh sửa hoặc bổ sung thông tin.

Quá trình phân công giám thị cho các ca thi cũng gặp nhiều khó khăn. Thông thường, nhà trường phải lập danh sách giám thị trên giấy, sau đó phân chia ca thi thủ công. Điều này dễ dẫn đến việc phân công không hợp lý, thiếu công bằng hoặc trùng lặp. Nếu cần thay đổi hoặc điều chỉnh lịch phân công, toàn bộ danh sách phải chỉnh sửa lại từ đầu, gây mất thời gian và dễ nhầm lẫn.

Ngoài ra, khi xảy ra các tình huống khẩn cấp như giám thị vắng mặt đột xuất, việc liên lạc để điều động người thay thế rất chậm trễ do không có hệ thống quản lý thông tin tập trung. Việc kiểm soát số lượng giám thị tham gia, theo dõi hiệu quả công việc hay đánh giá chất lượng giám sát cũng không được thực hiện hiệu quả.

Những hạn chế này không chỉ ảnh hưởng đến quá trình tổ chức thi mà còn làm giảm tính chuyên nghiệp và minh bạch trong công tác giám sát.

* + 1. Mô tả thông tin học phần

Trong môi trường giáo dục, việc quản lý thông tin học phần đóng vai trò quan trọng trong quá trình xây dựng kế hoạch giảng dạy và tổ chức học tập. Trước khi có hệ thống quản lý, thông tin về các học phần thường được lưu trữ rời rạc qua hồ sơ giấy hoặc bảng tính Excel. Các dữ liệu như mã học phần, tên học phần, số tín chỉ, khoa/bộ môn phụ trách thường xuyên bị thiếu sót hoặc không nhất quán, gây khó khăn cho việc tra cứu và sắp xếp lịch học.

Việc cập nhật thông tin học phần, đặc biệt là những thay đổi về mô tả nội dung, số tín chỉ hoặc lịch học, đòi hỏi sự kiểm tra thủ công trên nhiều tài liệu, dễ dẫn đến sai lệch hoặc chậm trễ. Điều này ảnh hưởng trực tiếp đến việc đăng ký học phần của sinh viên và công tác giảng dạy của giảng viên. Thêm vào đó, thông tin về học kỳ và năm học không được quản lý tập trung khiến việc xây dựng chương trình đào tạo trở nên kém linh hoạt và thiếu đồng bộ.

Ngoài ra, sự thiếu hụt trong việc mô tả chi tiết các học phần cũng gây khó khăn cho sinh viên khi lựa chọn môn học phù hợp với năng lực và định hướng của bản thân. Việc không có hệ thống phân loại rõ ràng theo khoa/bộ môn khiến công tác điều phối giảng dạy giữa các đơn vị gặp nhiều hạn chế.

* + 1. Mô tả thông tin phòng thi, ca thi

Việc tổ chức và quản lý thông tin liên quan đến phòng thi và ca thi thường được thực hiện thủ công, gây ra nhiều bất cập và khó khăn. Thông tin về phòng thi như mã phòng, địa chỉ và sức chứa tối đa thường được lưu trữ bằng giấy tờ hoặc bảng tính Excel. Điều này dễ dẫn đến sai sót trong việc phân bổ phòng thi, đặc biệt khi số lượng sinh viên tham gia kỳ thi đông hoặc có sự thay đổi đột xuất. Việc kiểm tra và cập nhật sức chứa phòng thi cũng không được thực hiện linh hoạt, gây khó khăn trong việc tối ưu hóa không gian thi.

Bên cạnh đó, công tác quản lý ca thi cũng gặp nhiều trở ngại. Thông tin về mã ca thi, học phần, phòng thi, giám thị, thời gian thi và danh sách sinh viên tham dự thường được ghi chép thủ công, dẫn đến tình trạng thiếu sót hoặc nhầm lẫn. Việc phân công giám thị, sắp xếp sinh viên vào các ca thi và phòng thi không được tối ưu, dễ gây trùng lặp hoặc bỏ sót. Khi cần điều chỉnh lịch thi hoặc thay đổi phòng thi, việc cập nhật thông tin rất phức tạp và tốn thời gian.

Ngoài ra, quá trình tổng hợp báo cáo sau mỗi ca thi thường được thực hiện bằng cách ghi chép tay hoặc lưu trữ trên các file rời rạc, khó kiểm soát và dễ thất lạc. Điều này ảnh hưởng trực tiếp đến việc đánh giá, tổng kết kết quả kỳ thi và gây khó khăn trong việc xử lý các sự cố phát sinh.

* + 1. Mô tả vận hành quy trình thi

Trong thực tế, quy trình tổ chức và vận hành một kỳ thi thường bao gồm nhiều bước và yêu cầu sự phối hợp chặt chẽ giữa các bộ phận liên quan như phòng đào tạo, giám thị, và sinh viên. Khi chưa có hệ thống quản lý tự động, toàn bộ quy trình này chủ yếu được thực hiện thủ công, dễ xảy ra sai sót và mất nhiều thời gian.

1. Chuẩn bị trước kỳ thi: Phòng đào tạo sẽ lên kế hoạc tổ chức kỳ thi, bao gồm việc lập danh sách sinh viên tham dự, phân bổ phòng thi, và sắp xếp lịch thi. Danh sách sinh viên thường được in ra giấy hoặc lưu trong file Excel, sau đó được chuyển cho giám thị để quản lý. Việc phân công giám thị, chuẩn bị đề thi và các tài liệu liên quan đều được thực hiện thủ công, dễ gây nhầm lẫn hoặc thiếu sót.

2. Tổ chức ca thi: Vào ngày thi, giám thị sẽ trực tiếp kiểm tra giấy tờ tùy thân của sinh viên (thẻ sinh viên, CMND/CCCD) để xác minh danh tính. Việc điểm danh được thực hiện bằng cách đánh dấu trên danh sách giấy, không đảm bảo tính chính xác tuyệt đối và dễ xảy ra gian lận. Nếu phát hiện sai sót, giám thị phải xử lý tình huống ngay tại chỗ mà không có sự hỗ trợ từ hệ thống.

3. Giám sát và kiểm soát quá trình thi: Trong suốt thời gian thi, giám thị chịu trách nhiệm giám sát phòng thi để đảm bảo trật tự và xử lý các tình huống bất thường. Tuy nhiên, việc theo dõi và ghi nhận các vi phạm quy chế thi chủ yếu dựa vào quan sát trực tiếp và ghi chép tay, dễ bỏ sót hoặc thiếu khách quan.

**4. Kết thúc và tổng hợp kết quả thi:** Sau khi kết thúc ca thi, giám thị thu bài và ký xác nhận vào biên bản. Việc tổng hợp biên bản, danh sách sinh viên dự thi và các tài liệu liên quan thường mất nhiều thời gian và dễ xảy ra thất lạc. Phòng đào tạo sẽ thu thập và kiểm tra lại thông tin trước khi tổng hợp báo cáo kết quả kỳ thi.

5. Báo cáo và lưu trữ: Các báo cáo về quá trình tổ chức thi, danh sách sinh viên tham dự và biên bản kiểm tra thường được lưu trữ dưới dạng giấy tờ hoặc file rời rạc, gây khó khăn trong việc tìm kiếm và tổng hợp thông tin khi cần thiết.

Quy trình vận hành kỳ thi theo cách thủ công này không chỉ tốn nhiều thời gian, công sức mà còn tiềm ẩn nhiều rủi ro như sai sót thông tin, gian lận trong thi cử và khó khăn trong việc xử lý các tình huống phát sinh. Do đó, việc áp dụng công nghệ vào quản lý và vận hành kỳ thi là xu hướng tất yếu, giúp tự động hóa quy trình, đảm bảo tính minh bạch, công bằng và nâng cao hiệu quả tổ chức kỳ thi.

* + 1. Giới thiệu thực tế quy trình triển khai, kiểm thử và bảo trì hệ thống

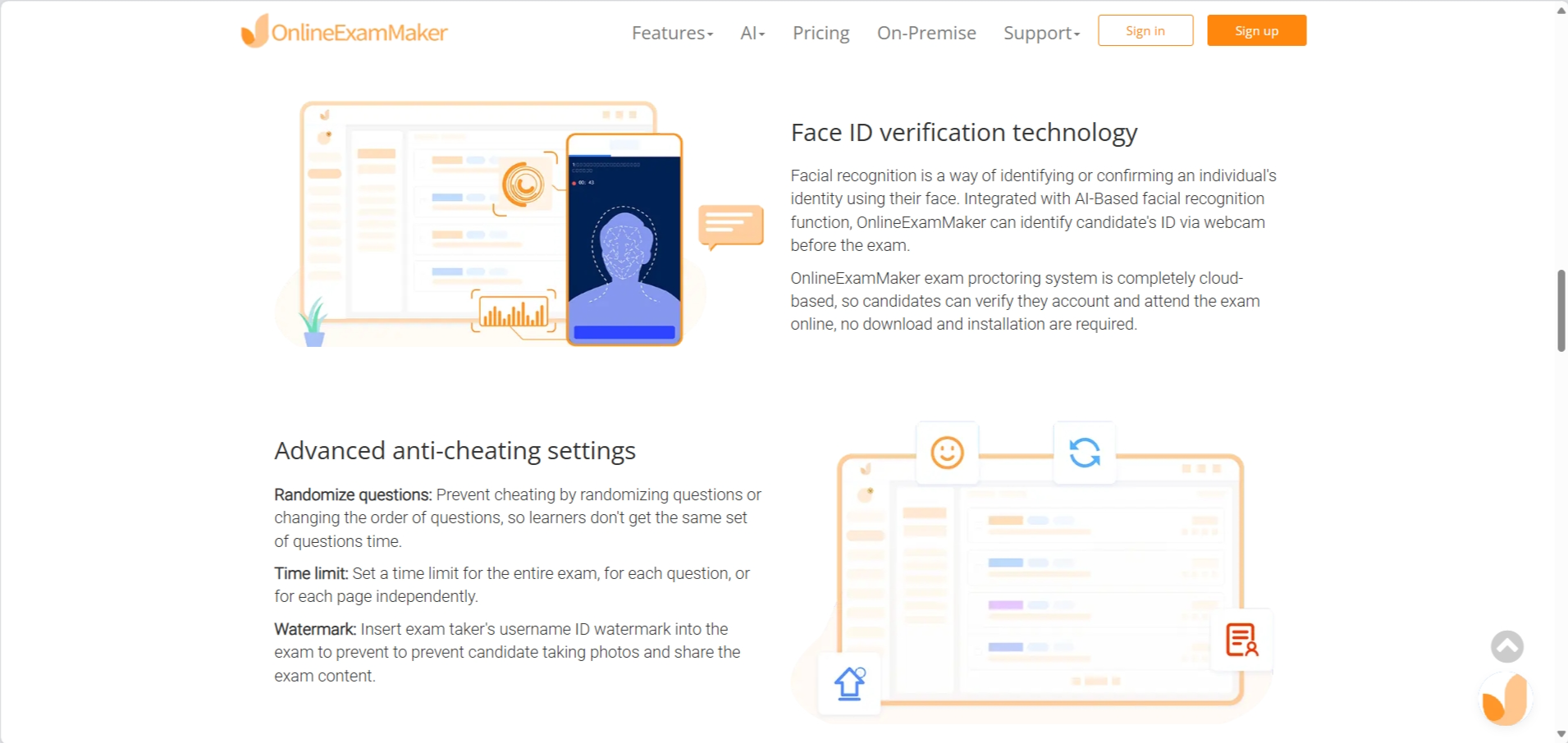
Sau khi nhóm đã chuẩn bị đầy đủ dữ liệu, công nghệ sử dụng và các yếu tố liên quan, chúng tôi nhận thấy việc áp dụng một mô hình làm việc linh hoạt, hiệu quả là rất cần thiết để triển khai, kiểm thử và bảo trì hệ thống. Quy trình này được thiết kế nhằm đảm bảo các giai đoạn phát triển hệ thống diễn ra một cách có tổ chức, từ việc phân chia nhiệm vụ, theo dõi tiến độ công việc đến kiểm soát thời gian và chất lượng. Thông qua việc triển khai từng bước nhỏ trong chu kỳ lặp lại, nhóm có thể vừa thực hiện vừa kiểm tra và cải tiến hệ thống. Cách tiếp cận này không chỉ giúp nhóm giám sát chặt chẽ tiến trình mà còn đảm bảo hệ thống luôn đáp ứng được các yêu cầu đề ra, phát hiện và xử lý lỗi một cách nhanh chóng. Nhờ đó, sản phẩm cuối cùng đạt được tính chính xác cao, tính linh hoạt và dễ dàng bảo trì trong quá trình sử dụng lâu dài.

* 1. Một số ứng dụng liên quan đến đề tài

Hệ thống nhận diện và xác minh danh tính đang ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực nhằm cải thiện hiệu quả quản lý, tăng cường an ninh và đảm bảo tính minh bạch. Đặc biệt, trong bối cảnh chuyển đổi số hiện nay việc ứng dụng công nghệ nhận diện khuôn mặt không chỉ mang lại sự tiện lợi mà còn giúp tối ưu hoá quy trình làm việc. Dưới đây là một số ứng dụng liên quan đến đề tài:

1) Hệ thống kiểm tra thi trực tuyến

Một trong những ứng dụng nổi bật của công nghệ nhận diện khuôn mặt là trong các hệ thống tổ chức thi trực tuyến. Ví dụ, trang web <https://onlineexammaker.com/> đã áp dụng nhận diện khuôn mặt để xác minh danh tính thí sinh ngay trước mỗi ca thi. Hệ thống này không chỉ giúp loại bỏ rủi ro gian lận danh tính mà còn nâng cao hiệu quả trong quản lý số lượng lớn thí sinh từ xa. Thông qua tính năng này, các trung tâm thi có thể tự động kiểm tra danh tính thí sinh, đối chiếu với cơ sở dữ liệu, và ghi nhận thời gian tham gia của từng cá nhân. Ngoài ra, việc kết hợp với các tính năng khác như giám sát bằng webcam hay kiểm soát hành vi cũng giúp tăng cường độ tin cậy và chính xác cho hệ thống thi trực tuyến.



Hình 1.1: Giao diện hệ thống kiểm tra thi trực tuyến

2) Hệ thống xác minh danh tính bằng công nghệ nhận diện khuôn mặt tại Vinpearl

Công nghệ nhận diện khuôn mặt cũng đã được ứng dụng thực tế trong lĩnh vực du lịch và dịch vụ khách sạn, điển hình là hệ thống xác minh danh tính tại Tổ hợp Vinpearl Nha Trang. Đây là một trong những đơn vị đầu tiên tại Việt Nam triển khai công nghệ này nhằm tối ưu hóa trải nghiệm của khách hàng. Hệ thống sử dụng các thiết bị nhận diện khuôn mặt hiện đại để xác thực danh tính của du khách khi ra vào các khu vực như khách sạn, khu nghỉ dưỡng, khu vui chơi giải trí VinWonders và nhà hàng. Thay vì các phương pháp kiểm soát truyền thống như xuất trình giấy tờ tùy thân hay thẻ ra vào, du khách chỉ cần quét khuôn mặt tại các cổng kiểm soát. Hệ thống tự động đối chiếu thông tin nhận diện với dữ liệu đã được lưu trữ trước đó để xác minh danh tính một cách nhanh chóng và chính xác.



Hình 1.2: Hình 1 ứng dụng xác minh khuôn mặt tại Vinpearl



Hình 1.3: Hình 2 ứng dụng xác minh khuôn mặt tại Vinpearl

* 1. Các yêu cầu cho hệ thống mới

Hệ thống nhận diện và xác minh danh tính sinh viên thông qua thẻ sinh viên và khuôn mặt cần phải đáp ứng một số yêu cầu chức năng và phi chức năng để đảm bảo hoạt động hiệu quả và ổn định. Dưới đây là các yêu cầu chi tiết cho hệ thống.

* + 1. Yêu cầu chức năng

|  |  |  |
| --- | --- | --- |
| **1. GIAO DIỆN QUẢN TRỊ** | | |
| **STT** | **Nội dung yêu cầu** | **Mô tả** |
| 1.1 | Đăng nhập hệ thống | Chức năng cho phép người quản trị đăng nhập vào hệ thống để xử lý thông tin theo các chức năng đã phân quyền |
| 1.2 | Quản lý danh sách sinh viên | Quản trị viên có thể xem, chỉnh sửa, xóa và thêm mới thông tin sinh viên. Chức năng này cho phép quản lý tất cả các thông tin liên quan đến sinh viên như mã sinh viên, họ tên, khóa học, ngành học, thông tin liên hệ, trạng thái học tập, ảnh thẻ và ảnh khuôn mặt. |
| 1.3 | Quản lý danh sách giám thị | Quản trị viên có thể xem, thêm mới, chỉnh sửa và xóa thông tin giám thị. Chức năng này cho phép phân công giám thị cho các ca thi và phòng thi. |
| 1.4 | Quản lý danh sách học phần | Quản trị viên có thể xem, chỉnh sửa và thêm mới các học phần. Chức năng này bao gồm mã học phần, tên học phần, số tín chỉ, khoa/bộ môn, học kỳ và năm học. |
| 1.5 | Quản lý danh sách phòng thi | Quản trị viên có thể xem, chỉnh sửa, thêm mới và xóa thông tin các phòng thi. Các thông tin quản lý bao gồm mã phòng, địa chỉ phòng thi, và sức chứa tối đa của phòng. |
| 1.6 | Quản lý danh sách ca thi | Quản trị viên có thể xem, chỉnh sửa và thêm mới các ca thi, bao gồm thông tin về học phần, phòng thi, giám thị, lịch thi và trạng thái ca thi. Các báo cáo về kỳ thi cũng có thể được truy xuất và tải xuống. |
| **2. GIAO DIỆN GIÁM THỊ** | | |
| 2.1 | Quản lý thông tin cá nhân | Giám thị có thể xem và chỉnh sửa thông tin cá nhân của mình, bao gồm tên, email, số điện thoại và khoa/bộ môn. |
| 2.2 | Quản lý lịch coi thi | Giám thị có thể xem lịch thi của mình, bao gồm thông tin về các ca thi mà mình được phân công, phòng thi và giám sát sinh viên. |
| 2.3 | Bắt đầu ca thi | Giám thị có thể bắt đầu một ca thi bằng cách nhập mã ca thi. Hệ thống sẽ xác minh thông tin sinh viên, giám sát quá trình thi, và cung cấp các công cụ hỗ trợ trong suốt kỳ thi. |
| **3. GIAO DIỆN SINH VIÊN** | | |
| 3.1 | Quản lý thông tin cá nhân | Sinh viên có thể xem và cập nhật các thông tin cá nhân của mình, bao gồm họ tên, địa chỉ email, số điện thoại và ảnh thẻ. |
| 3.2 | Quản lý lịch thi | Sinh viên có thể xem lịch thi của mình, bao gồm các thông tin về học phần, phòng thi, thời gian thi và tình trạng đăng ký thi. |

Bảng 1.3: Bảng mô tả yêu cầu chức năng hệ thống

* + 1. Yêu cầu phi chức năng

|  |  |  |
| --- | --- | --- |
| **STT** | **Nội dung yêu cầu** | **Mô tả** |
| 1 | Hệ thống phải xử lý nhanh chóng và hiệu quả. | Đảm bảo trích xuất và xác minh thông tin nhanh chóng, Tránh gây trì hoãn trong các quy trình như tham gia thi. |
| 2 | Hệ thống phải có khả năng mở rộng. | Đáp ứng nhu cầu trong tương lai như số lượng sinh viên tăng hoặc tích hợp tính năng mới. |
| 3 | Hệ thống phải đảm bảo tính bảo mật cao. | Mã hóa các thông tin nhạy cảm như mã số sinh viên, thông tin cá nhân. Quy trình truy cập được bảo mật, chỉ người có quyền mới truy cập, chỉnh sửa dữ liệu. |
| 4 | Hệ thống phải đảm bảo tính khả dụng và độ tin cậy. | Luôn hoạt động ổn định trong quá trình triển khai. Có khả năng tự động khôi phục sau sự cố và tránh mất dữ liệu quan trọng. |
| 5 | Hệ thống phải tương thích với các thiết bị đầu vào và nền tảng khác nhau. | Hỗ trợ máy ảnh, webcam và các thiết bị nhận diện khuôn mặt. Hỗ trợ trên nhiều nền tảng như máy tính và thiết bị di động. |
| 6 | Hệ thống phải dễ bảo trì và cập nhật. | Dễ dàng nâng cấp phần mềm, triển khai tính năng mới hoặc sửa lỗi mà không làm gián đoạn hoạt động. |
| 7 | Hệ thống phải tương tác tốt với các hệ thống khác. | Tích hợp với hệ thống quản lý sinh viên, hệ thống thi cử. API phải được thiết kế để tương tác được với hệ thống bên ngoài. |

Bảng 1.4: Bảng mô tả yêu cầu phi chức năng hệ thống

1. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN

Chương 2 sẽ tập trung vào việc phân tích và thiết kế hệ thống vận hành và quản lý quy trình thi trắc nghiệm khách quan, nhằm hướng đến việc xây dựng một giải pháp hiện đại, hiệu quả, giảm thiểu các sai sót và tối ưu hóa quy trình kiểm tra, giám sát, cũng như đánh giá trong giáo dục. Nội dung chương được trình bày chi tiết qua ba phần chính: phân tích yêu cầu, thiết kế hệ thống, và xác định các yếu tố cần thiết để phát triển một hệ hoàn chỉnh. Quá trình phát triển hệ thống sẽ áp dụng theo phương pháp mô hình Scrum, trong đó công việc được chia thành các nhiệm vụ nhỏ, triển khai theo từng giai đoạn (sprint), đảm bảo linh hoạt trong quá trình xây dựng và thích ứng kịp thời với những yêu cầu phát sinh. Thông qua đó chương sẽ cung cấp cái nhìn tổng quan về cách thức triển khai hoàn thiện hệ thống để xử lý các yêu cầu đã đặt ra.

* 1. Phân tích yêu cầu và xác định tác nhân
     1. Phân tích yêu cầu hệ thống

Mục tiêu của phân tích yêu cầu hệ thống là xác định các tính năng và yêu cầu mà hệ thống phải đáp ứng trong quá trình phát triển. Để đảm bảo tính hoàn chỉnh, hiệu quả và khả năng mở rộng của hệ thống, các yêu cầu được phân chia thành các Sprint cụ thể, mỗi Sprint có mục đích và yêu cầu rõ ràng. Dưới đây là mô tả tổng thể các yêu cầu hệ thống được chia theo từng Sprint:

|  |  |  |
| --- | --- | --- |
| **Sprint** | **Nội dung** | **Mục đích hướng đến** |
| 1 | Quản lý thông tin sinh viên | Tạo cơ sở dữ liệu và giao diện quản lý thông tin sinh viên, bao gồm việc lưu trữ, chỉnh sửa và tra cứu thông tin sinh viên như mã số, họ tên, khóa học, ảnh thẻ và ảnh khuôn mặt. |
| 2 | Quản lý thông tin giám thị | Tạo hệ thống quản lý thông tin giám thị, bao gồm việc thêm, sửa, xóa, và phân công giám thị cho các ca thi, phòng thi. |
| 3 | Quản lý học phần | Xây dựng tính năng quản lý học phần, cho phép nhập, sửa, xóa các học phần trong hệ thống, bao gồm các thông tin như tên học phần, mã học phần, số tín chỉ, khoa/bộ môn. |
| 4 | Quản lý phòng thi, ca thi | Quản lý thông tin phòng thi và ca thi, bao gồm việc tạo phòng thi, phân bổ phòng thi cho các ca thi, đồng thời đảm bảo thông tin phòng thi và ca thi được cập nhật chính xác. |
| 5 | Quản lý quy trình thi | Xây dựng hệ thống quản lý quy trình thi, bao gồm việc tổ chức và giám sát các kỳ thi, từ việc xác minh sinh viên, điểm danh, đến việc xử lý kết quả thi, báo cáo kết quả và theo dõi tiến độ thi. |

Bảng 2.1: Bảng tóm tắt Sprint và mục tiêu hướng đến

Thông qua các Sprint này, hệ thống sẽ dần được xây dựng từ các tính năng cơ bản đến các tính năng phức tạp hơn, đồng thời đảm bảo chất lượng và hiệu quả hoạt động trong môi trường thực tế khi triển khai bài toán.

* + 1. Xác định các tác nhân tác động lên hệ thống

Hệ thống vận hành và quản lý quy trình thi trắc nghiệm khách quan xác định ba tác nhân chính là *Sinh viên*, *Giám thị* và *Quản trị viên*. Mỗi tác nhân có vai trò và trách nhiệm cụ thể trong việc đảm bảo hệ thống vận hành trơn tru, hiệu quả và minh bạch.

* + - 1. Tác nhân sinh viên

Sinh viên là đối tượng chính tham gia vào các kỳ thi và thực hiện các nhiệm vụ liên quan trong hệ thống. Trách nhiệm của sinh viên bao gồm việc cung cấp đầy đủ và chính xác thông tin cá nhân, tuân thủ quy trình xác minh danh tính trước mỗi ca thi và tham gia thi theo các yêu cầu đặt ra.

* + - 1. Tác nhân giám thị

Giám thị là tác nhân hỗ trợ và giám sát quá trình thi, đóng vai trò quan trọng trong việc đảm bảo sự trung thực và minh bạch trong suốt kỳ thi. Giám thị đăng nhập vào hệ thống qua tài khoản được cấp và thực hiện nhiều chức năng như kiểm tra danh sách sinh viên tham gia ca thi, giám sát quy trình xác minh danh tính tự động và hỗ trợ theo dõi nhằm phát hiện các hành vi gian lận trong phòng thi. Ngoài ra, giám thị còn chịu trách nhiệm ghi nhận sự cố hoặc vi phạm xảy ra trong kỳ thi và báo cáo kết quả sau khi ca thi kết thúc. Vai trò của giám thị giúp đảm bảo các quy trình thi cử được diễn ra một cách chặt chẽ và nghiêm túc.

* + - 1. Tác nhân quản trị viên

Quản trị viên là tác nhân thiết yếu phải có trong hệ thống, chịu trách nhiệm quản lý và điều phối toàn bộ các hoạt động nhằm đảm bảo hệ thống vận hành ổn định, dữ liệu được bảo mật và các yêu cầu người dùng được đáp ứng. Với quyền hạn cao nhất, quản trị viên thực hiện nhiều chức năng quan trọng như quản lý người dùng, bao gồm tạo, chỉnh sửa hoặc xóa tài khoản của giám thị và sinh viên, đảm bảo quyền truy cập hợp lệ và hợp lý trong hệ thống. Bên cạnh đó, quản trị viên có thể tùy chỉnh các tham số hệ thống như quy trình thi, thông tin kỳ thi và cài đặt bảo mật để đáp ứng nhu cầu sử dụng và đảm bảo tính linh hoạt của hệ thống. Ngoài ra, họ còn chịu trách nhiệm theo dõi toàn bộ quá trình thi, bao gồm thời gian tham gia của sinh viên, trạng thái nhận diện khuôn mặt và hoạt động của giám thị, nhằm phát hiện và xử lý kịp thời các bất thường. Để duy trì tính toàn vẹn của thông tin, quản trị viên quản lý cơ sở dữ liệu thông qua các hoạt động như sao lưu, phục hồi và bảo mật dữ liệu, ngăn ngừa mất mát hoặc rò rỉ thông tin. Trong trường hợp phát sinh sự cố, quản trị viên tiếp nhận các báo lỗi hoặc khiếu nại từ giám thị và sinh viên, xử lý nhanh chóng để đảm bảo hệ thống hoạt động liên tục và ổn định. Với vai trò điều hành và giám sát toàn diện, quản trị viên không chỉ đảm bảo rằng mọi quy trình trong hệ thống diễn ra hiệu quả và minh bạch, mà còn đóng vai trò quyết định trong việc nâng cao chất lượng vận hành và hiệu quả tổng thể của hệ thống.

* 1. Sprint 1: Quản lý thông tin cá nhân sinh viên
     1. Giới thiệu Sprint 1

Sprint 1 của dự án sẽ tập trung vào việc xây dựng hệ thống quản lý thông tin sinh viên, đặc biệt là quy trình xác minh danh tính sinh viên thông qua ảnh thẻ sinh viên. Trong giai đoạn này, hệ thống sẽ nhận diện và xử lý ảnh thẻ mà sinh viên tải lên. Quá trình bắt đầu khi giám thị yêu cầu sinh viên chụp ảnh thẻ và tải lên hệ thống. Sau khi tiếp nhận, hệ thống sẽ thực hiện các bước tiền xử lý ảnh, bao gồm các thao tác như cắt ảnh, điều chỉnh độ sáng, độ tương phản và làm thẳng ảnh để đảm bảo hình ảnh rõ ràng và dễ dàng xử lý.

Tiếp theo, công nghệ nhận dạng ký tự quang học (OCR) sẽ được sử dụng để trích xuất các thông tin quan trọng từ thẻ, bao gồm mã số sinh viên, họ tên, ngày sinh, ngành học, khóa học, và đơn vị đào tạo. Các thông tin này sẽ được lưu trữ vào cơ sở dữ liệu để sử dụng cho các bước xác thực danh tính trong các quy trình tiếp theo.

Sprint 1 dự kiến sẽ kéo dài trong 11 ngày, từ ngày 30/9/2024 đến 13/10/2024. Các công việc chính trong sprint này bao gồm: thu thập và chuẩn bị dữ liệu mẫu, xây dựng chức năng trích xuất thông tin từ ảnh thẻ sinh viên, huấn luyện mô hình nhận dạng Yolo v11 với tập dữ liệu THE SINH VIEN, thiết kế giao diện người dùng, và tích hợp chức năng đối chiếu thông tin với hệ thống backend và cơ sở dữ liệu sinh viên. Cuối cùng, các mô hình và API sẽ được kiểm thử để đảm bảo tính chính xác và hiệu quả trong việc trích xuất thông tin từ ảnh thẻ sinh viên.

* + 1. Thiết kế dữ liệu sinh viên

Để lưu trữ thông tin sinh viên như đã đề cập trong Chương 1, phần này chỉ trình bày chi tiết về thiết kế bảng dữ liệu nhằm đáp ứng yêu cầu lưu trữ và quản lý hiệu quả. Bảng dữ liệu được xây dựng dựa trên việc phân tích các thông tin của sinh viên, bổ sung thêm một số trường thông tin cần thiết cho quá trình xây dựng hệ thống sau này. Các thông tin cụ thể về cấu trúc và nội dung của bảng dữ liệu lưu trữ thông tin sinh viên được mô tả chi tiết trong **Bảng 2.1.**

*Mục đích*: Lưu trữ các thông tin của sinh viên.

*Tên bảng*: tbl Sinh viên

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Diễn giải** | **Tên trường** | **Kiểu** | **Mặc định** | **Ràng buộc** | **Ghi chú** |
| 1 | Tự động tăng | student\_id | \_id |  | PK |  |
| 2 | Mã sinh viên | student\_code | Text | Unique |  |  |
| 3 | Họ tên sinh viên | full\_name | Text |  |  |  |
| 4 | Ngày sinh | date\_of\_birth | Date |  |  |  |
| 5 | Lớp | class\_name | Text |  |  |  |
| 6 | Ngành | department | Text |  |  |  |
| 7 | Đơn vị | course\_name | Text |  |  |  |
| 8 | Email | email | Text |  |  |  |
| 9 | Số điện thoại | phone\_number | Text |  |  |  |
| 10 | Kết quả thi | exam\_result | Text |  | FK |  |
| 11 | Trạng thái | status | Boolean | 1 |  | Mô tả tình trạng của sinh viên(đang học/đã nghỉ) |
| 12 | Lưu ảnh thẻ | student\_photo | Text |  |  |  |
| 13 | Lưu khuôn mặt trên thẻ | face\_photo | Text |  |  |  |

Bảng 2.2: Bảng mô tả dữ liệu lưu trữ trong tbl sinh viên

* + 1. Quy trình quản lý thông tin sinh viên

Quy trình quản lý thông tin sinh viên được thiết kế để vận hành tự động và tuần tự nhằm đảm bảo độ chính xác và tối ưu thời gian thực hiện. Với kết quả mong muốn là đọc và lưu trữ được các thông tin sinh viên, hệ thống sẽ tiến hành các bước với chi tiết như sau:

|  |  |
| --- | --- |
| **Mục đích** | Sinh viên cập nhật được thông tin đặc biệt là ảnh thẻ và ảnh khuôn mặt lên hệ thống |
| **Các bước thực hiện** | 1. Đăng nhập vào hệ thống: Sinh viên đăng nhập vào hệ thống bằng mã sinh viên và mật khẩu.  2. Truy cập giao diện cập nhật thông tin: Sau khi đăng nhập, sinh viên vào mục "Cập nhật thông tin cá nhân".  3. Chọn ảnh thẻ và ảnh khuôn mặt: Sinh viên chọn và tải lên ảnh thẻ sinh viên và ảnh khuôn mặt từ thiết bị của mình. Hệ thống sẽ yêu cầu xác minh định dạng ảnh (ví dụ: JPG, PNG) và kích thước phù hợp.  4. Cập nhật thông tin cá nhân: Sinh viên có thể sửa các thông tin cá nhân khác như số điện thoại, email, lớp học, khoa/viện, v.v. (Nếu cần).  5. Lưu thông tin: Sau khi hoàn tất cập nhật, sinh viên bấm "Lưu" để cập nhật thông tin vào hệ thống. Hệ thống kiểm tra tính hợp lệ của thông tin và lưu vào cơ sở dữ liệu.  6. Thông báo kết quả: Sau khi thông tin được lưu thành công, hệ thống thông báo "Cập nhật thành công". Nếu có lỗi (ví dụ: ảnh không hợp lệ, thiếu thông tin bắt buộc), hệ thống sẽ thông báo lỗi và yêu cầu sinh viên chỉnh sửa. |
| **Các bước**  **bổ sung** | 1. Kiểm tra định dạng ảnh: Hệ thống kiểm tra ảnh thẻ và ảnh khuôn mặt xem có đúng định dạng và kích thước yêu cầu không.  2. Xác thực thông tin đã thay đổi: Sau khi cập nhật, hệ thống gửi thông báo yêu cầu sinh viên xác nhận lại thông tin đã thay đổi. |
| **Các ngoại lệ** | 1. Ảnh thẻ không hợp lệ: Nếu ảnh thẻ không đạt yêu cầu (định dạng sai, kích thước quá lớn hoặc quá nhỏ), hệ thống sẽ yêu cầu sinh viên tải lại ảnh đúng.  2. Thông tin không đầy đủ: Nếu sinh viên không điền đủ thông tin bắt buộc (ví dụ: không nhập số điện thoại, email), hệ thống sẽ thông báo và yêu cầu cập nhật lại.  3. Lỗi kết nối hệ thống: Nếu xảy ra sự cố trong quá trình tải lên hoặc lưu thông tin, hệ thống sẽ thông báo lỗi và yêu cầu thử lại sau. |
| **Tài liệu liên quan** | 1. Hướng dẫn sử dụng hệ thống cho sinh viên.  2. Quy định về việc tải ảnh thẻ sinh viên và ảnh khuôn mặt.  3. Tài liệu kỹ thuật về việc xử lý ảnh và định dạng ảnh trong hệ thống. |
| **Tham chiếu** | 1. Bảng thông tin sinh viên: Để tham chiếu thông tin cá nhân của sinh viên và ảnh đã lưu trong cơ sở dữ liệu.  2. API xác minh ảnh khuôn mặt: Để tham chiếu vào quy trình nhận diện và xác minh khuôn mặt khi sinh viên tham gia kỳ thi. |

Bảng 2.3: Bảng chức năng quản lý thông tin sinh viên

* + 1. Thiết kế quy trình nghiệp vụ

Quy trình nghiệp vụ mô tả các bước liên quan đến việc sinh viên cập nhật thông tin cá nhân, đặc biệt là ảnh thẻ và ảnh khuôn mặt, trên hệ thống. Quy trình này đảm bảo rằng thông tin được lưu trữ chính xác, giúp dễ dàng quản lý và xác minh danh tính sinh viên khi tham gia các kỳ thi.

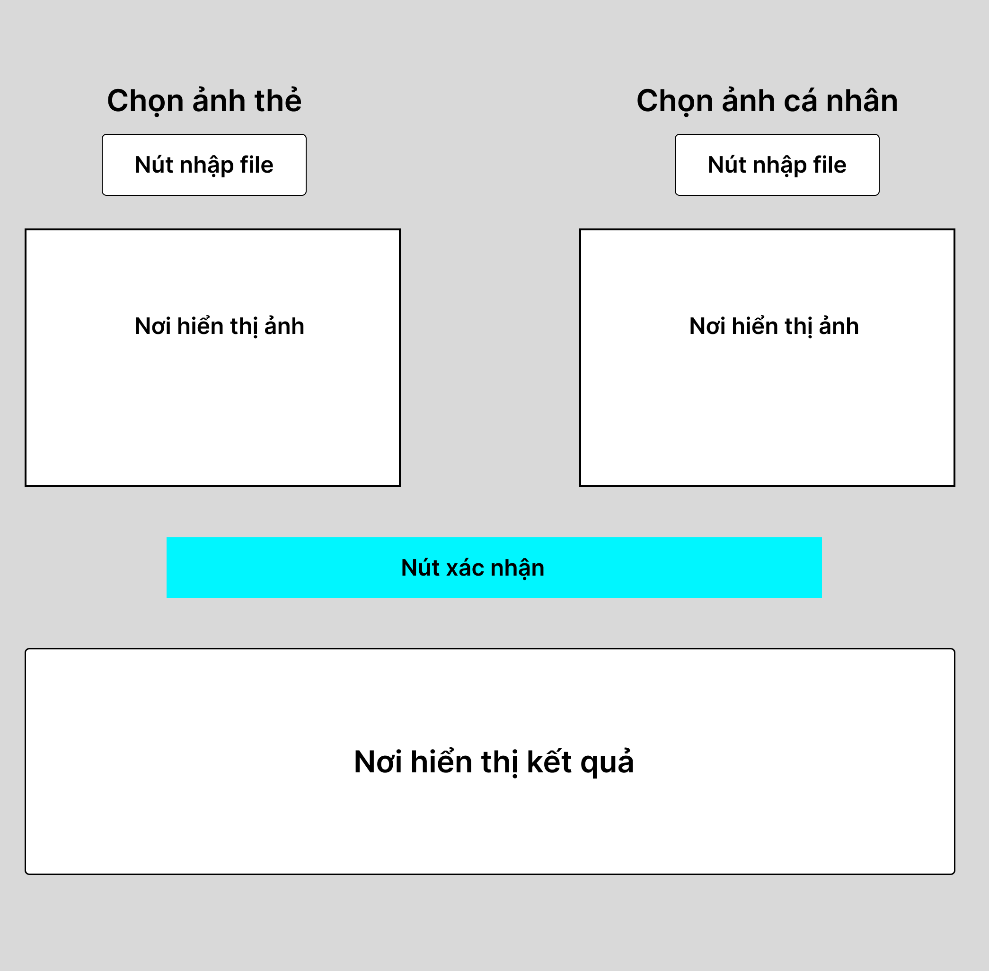
A diagram with text and black text

Description automatically generated

Hình 2.1: Biểu đồ tuần tự chức năng cập nhật hình ảnh sinh viên

* + 1. Thiết kế giao diện

**Hình 2.2** mô tả giao diện mẫu của chức năng đọc danh sách sinh viên trong hệ thống. Giao diện bao gồm hai mục chính: nơi tiếp nhận ảnh thẻ sinh viên và ảnh thẻ thực tế. Ảnh thẻ sinh viên được tải lên sẽ trải qua quá trình xử lý. Kết quả sau khi xử lý sẽ được hiển thị tại khu vực "Nơi hiển thị kết quả", giúp giám thị dễ dàng theo dõi và xác minh thông tin của sinh viên.



Hình 2.2: Mẫu giao diện chức năng cập nhật hình ảnh sinh viên

* 1. Sprint 2: Chức năng so khớp thông tin sinh viên với danh sách trong phòng thi
     1. Giới thiệu Sprint 2

Sau khi hoàn tất Mô-đun trích xuất thông tin trên thẻ sinh viên, bước tiếp theo là xây dựng chức năng hiển thị danh sách sinh viên trong phòng thi kết hợp với Mô-đun xác nhận danh tính sinh viên để hỗ trợ giám thị trong việc kiểm tra thông tin. Giám thị sẽ đăng nhập vào hệ thống, tìm kiếm phòng thi được phân công và hệ thống sẽ hiển thị danh sách sinh viên tương ứng. Khi thông tin đã được xác nhận chính xác, giám thị có thể trích xuất danh sách sinh viên thành file Excel và nhập vào giao diện kiểm tra khuôn mặt kết hợp với trích xuất thông tin thẻ sinh viên để kiểm tra từng sinh viên một cách cụ thể. Chức năng này được triển khai trong vòng 2 tuần từ 13/10/2024 đến 27/10/2024, bao gồm các công việc như chuẩn bị danh sách thí sinh, xây dựng chức năng đối chiếu danh sách sinh viên trong phòng thi, thêm, sửa, xóa thông tin sinh viên, xem danh sách sinh viên và xuất kết quả ra file Excel, cùng với việc kiểm thử chức năng đối chiếu và CRUD dữ liệu vào cơ sở dữ liệu. Việc hoàn thành chức năng này giúp quy trình kiểm tra và xác minh thông tin sinh viên trong phòng thi diễn ra thuận tiện, chính xác và hiệu quả hơn.

A diagram of a diagram

Description automatically generated

Hình 2.3: Biểu đồ ca sử dụng của giám thị trong chức năng so khớp thông tin

* + 1. Thiết kế dữ liệu phòng thi

Để hỗ trợ việc lưu trữ và quản lý thông tin sinh viên trong quá trình xác minh danh tính, phần này sẽ trình bày chi tiết về thiết kế bảng dữ liệu phục vụ cho việc quản lý phòng thi. Bảng dữ liệu được thiết kế dựa trên việc phân tích các thông tin liên quan đến phòng thi và sinh viên, đồng thời bổ sung một số trường thông tin quan trọng nhằm đảm bảo hệ thống vận hành hiệu quả và đáp ứng các yêu cầu mở rộng trong tương lai. Cấu trúc và nội dung cụ thể của bảng dữ liệu lưu trữ thông tin phòng thi sẽ được trình bày chi tiết trong **Bảng 2.4**.

*Mục đích*: Lưu trữ các thông tin của phòng thi

*Tên bảng*: tbl ExamSessions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Diễn giải** | **Tên trường** | **Kiểu** | **Mặc định** | **Ràng buộc** | **Ghi chú** |
| 1 | Mã ca thi | session\_code | Text |  | PK | Khoá chính |
| 2 | Mã học phần | course\_code | Text |  | FK | Khoá ngoại với tbl Courses |
| 3 | Mã phòng thi | room\_code | Text |  | FK | Khoá ngoại với tbl Rooms |
| 4 | Danh sách mã sinh viên | student\_codes | Text |  | FK | Khoá ngoại với tbl Students  Chứa mã sinh viên có trong phòng thi |
| 5 | Mã giám thị | proctor\_code | Text |  | FK | Khoá ngoại với tbl Rooms |
| 6 | Thời điểm thi | exam\_date\_time | Date |  |  |  |
| 7 | Trạng thái ca thi | session\_status | Text |  |  |  |
| 8 | Đường dẫn file báo cáo | report\_file\_path |  |  |  |  |

Bảng 2.4: Bảng mô tả dữ liệu lưu trữ trong tbl ExamSessions

* + 1. Quy trình đối chiếu thông tin sinh viên với danh sách phòng thi

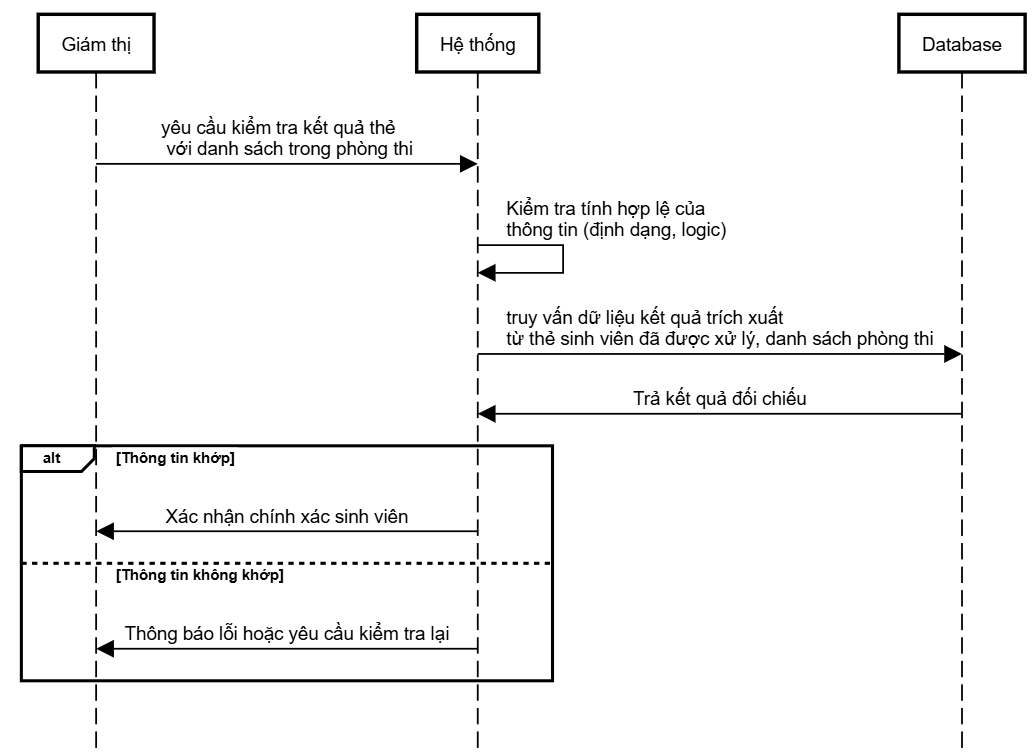
Quy trình đối chiếu thông tin sinh viên với danh sách phòng thi được thiết kế để vận hành tự động, tuần tự nhằm đảm bảo độ chính xác và tối ưu thời gian thực hiện. Sau khi đã có thông tin trích xuất từ thẻ, hệ thống sẽ tiến hành so khớp với danh sách sinh viên được giám thị nhập vào. Với kết quả mong muốn là xác nhận sự có mặt của sinh viên trong phòng thi, hệ thống sẽ tiến hành các bước với chi tiết như sau:

|  |  |
| --- | --- |
| **Mục đích** | Xác nhận sinh viên có tên trong danh sách phòng thi |
| **Các bước thực hiện** | 1. Giám thị đăng nhập vào hệ thống.  2. Giám thị tìm phòng thi.  3. Hệ thống hiển thị danh sách sinh viên.  4. Giám thị kiểm tra danh sách và xác nhận sinh viên.  5. Giám thị cập nhật trạng thái và xuất danh sách.  6. Thông báo sinh viên không có tên trong danh sách. |
| **Các bước**  **bổ sung** | - Tìm kiếm sinh viên trong danh sách.  - Cập nhật thông tin sinh viên nếu có sai sót. |
| **Các ngoại lệ** | - Sinh viên không có tên trong danh sách.  - Không xác minh được thông tin sinh viên (thẻ sinh viên/khuôn mặt). |
| **Tài liệu liên quan** | - Hướng dẫn sử dụng hệ thống.  - Biểu mẫu danh sách sinh viên. |
| **Tham chiếu** | - Quy trình xác nhận danh tính sinh viên.  - Quy trình quản lý thông tin sinh viên. |

Bảng 2.5: Bảng chức năng đối chiếu thông tin sinh viên với danh sách phòng thi

* + 1. Thiết kế quy trình nghiệp vụ

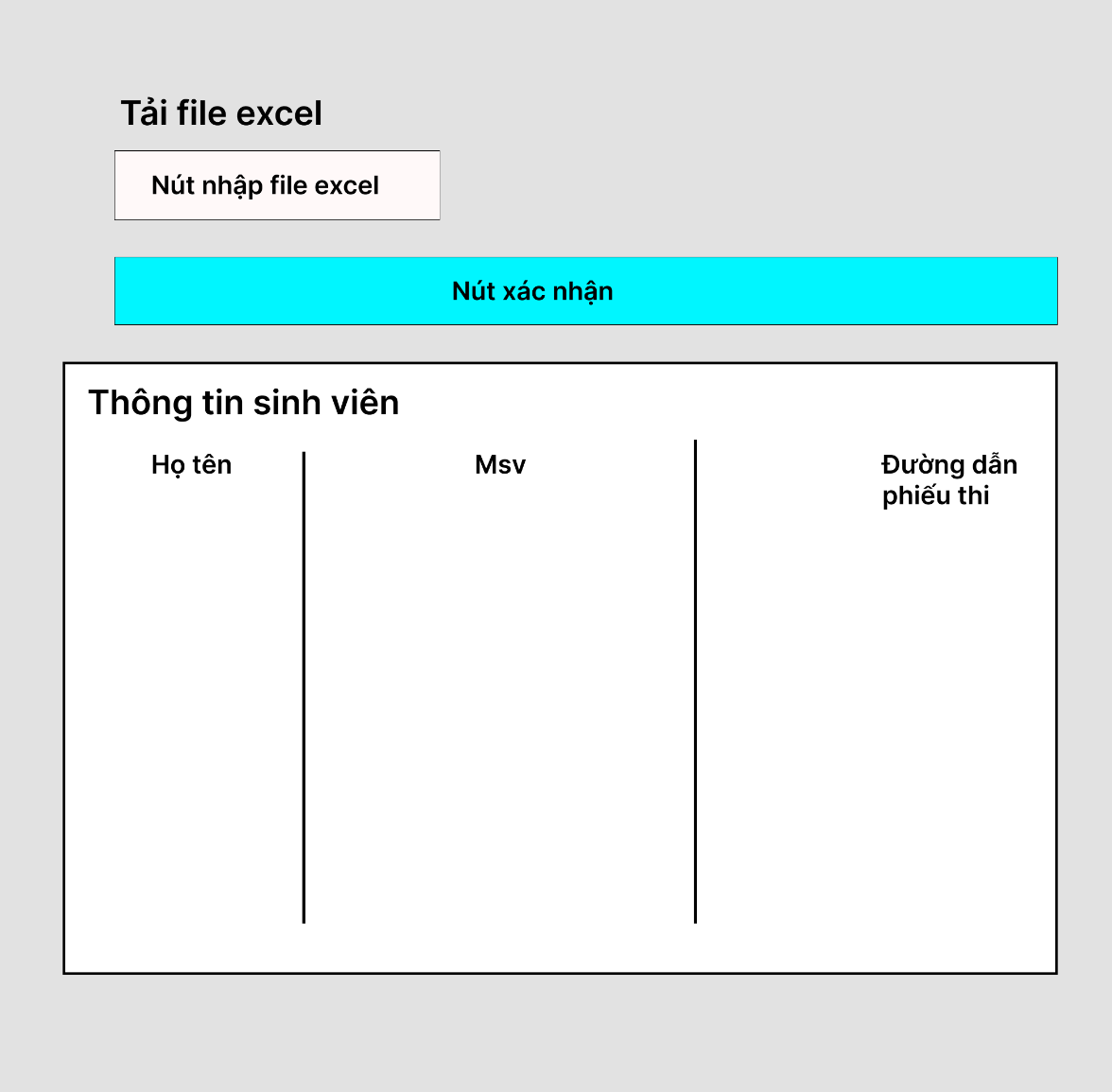
Quy trình đối chiếu thông tin sinh viên với danh sách trong phòng thi bắt đầu bằng việc giám thị truy cập vào hệ thống để lấy danh sách sinh viên của phòng thi mà mình được phân công. Sau khi đã hiển thị danh sách, giám thị tiến hành tải về file excel để sau nhập vào tính năng xác minh, hệ thống sẽ chỉ lưu trữ tạm thông tin cần thiết để xác minh danh tính như *mã số sinh viên*, *họ tên sinh viên*. Kết hợp với thông tin đã đọc được từ trước quy trình đối chiếu được triển khai như sau:



Hình 2.4: Biểu đồ tuần tự chức năng đối chiếu thông tin với danh sách trong phòng thi

* + 1. Thiết kế giao diện

**Hình 2.5** mô tả giao diện mẫu của chức năng đối chiếu thông tin sinh viên với danh sách trong phòng thi. Giao diện bao gồm nút tải lên file excel và nơi để hiển thị kết quả đối chiếu.



Hình 2.5: Mẫu giao diện chức năng đối chiếu với các sinh viên trong phòng thi

* 1. Sprint 3: Đối chiếu khuôn mặt sinh viên trong cơ sở dữ liệu với ảnh thực tế
     1. Giới thiệu Sprint 3

Sprint 3 của đề tài tập trung vào việc phát triển chức năng đối chiếu khuôn mặt trên thẻ sinh viên với khuôn mặt thực tế trong quá trình xác minh danh tính. Đây là bước quan trọng nhằm đảm bảo tính chính xác và độ tin cậy của hệ thống trong việc nhận diện và xác thực sinh viên tham gia kỳ thi. Trong thời gian 12 ngày, từ 27/10/2024 đến 8/11/2024, các công việc chính được thực hiện bao gồm: thu thập và chuẩn bị dữ liệu mẫu, xây dựng chức năng chính, tạo giao diện và tích hợp chức năng đối chiếu qua backend, truy vấn cơ sở dữ liệu, và kiểm thử chức năng đối chiếu khuôn mặt. Sprint này không chỉ hoàn thiện chức năng đối chiếu mà còn tích hợp giao diện người dùng, đảm bảo tính liền mạch và hiệu quả trong quá trình sử dụng.

A diagram with black text and black circles

Description automatically generated

Hình 2.6: Biểu đồ ca sử dụng đối chiếu khuôn mặt của giám thị

A diagram of a person

Description automatically generated

Hình 2.7: Biểu đồ ca sử dụng đối chiếu khuôn mặt của sinh viên

* + 1. Quy trình đối chiếu khuôn mặt

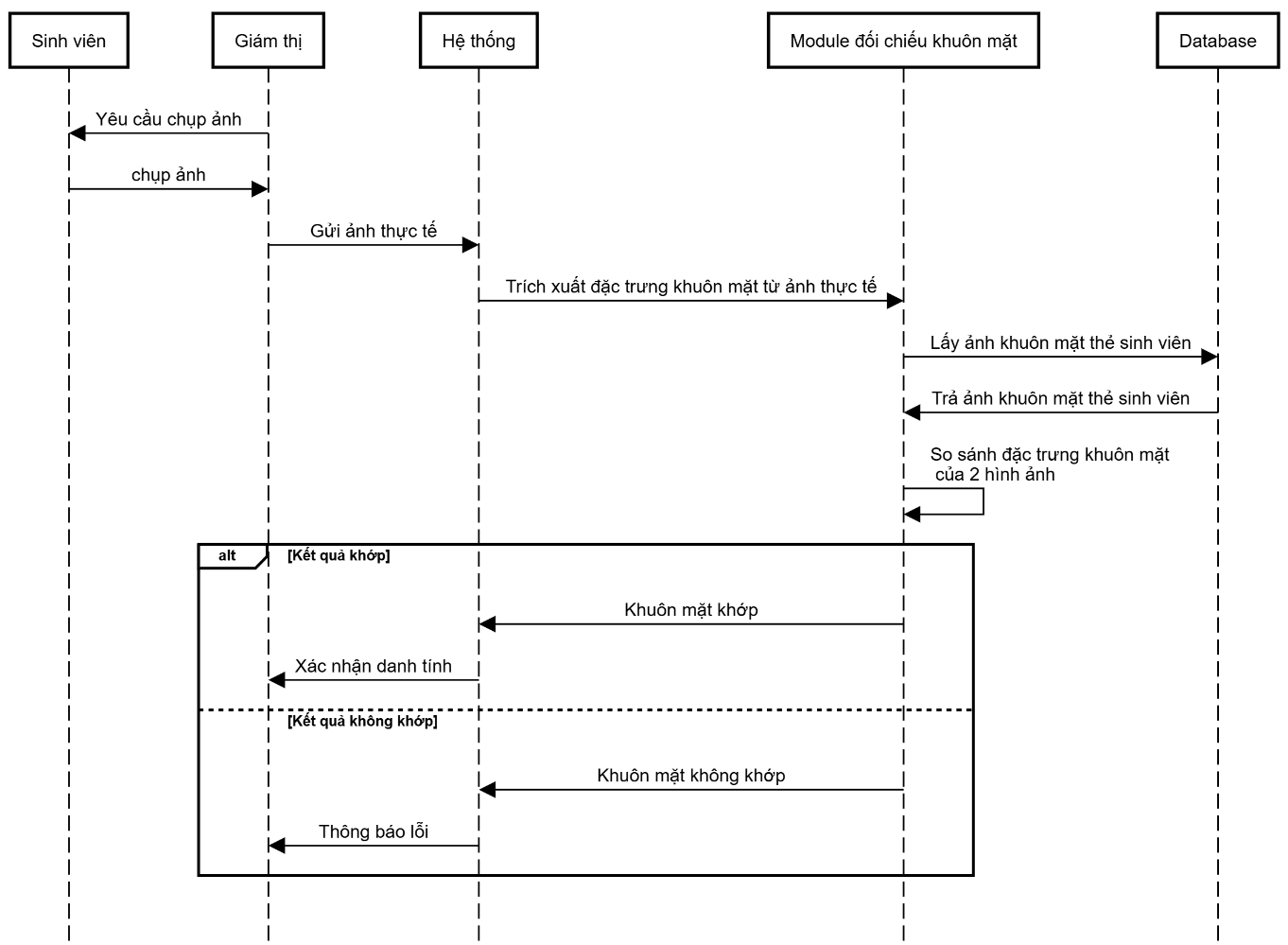
Quy trình thực hiện chức năng đối chiếu khuôn mặt bao gồm ba bước chính. Bước đầu tiên là chuẩn bị dữ liệu đầu vào, trong đó đảm bảo rằng hai hình ảnh quan trọng, bao gồm ảnh sinh viên trên thẻ và ảnh sinh viên thực tế, phải được cắt đúng khuôn mặt và chính xác để sẵn sàng cho các bước tiếp theo. Bước thứ hai là xây dựng chương trình đối chiếu khuôn mặt. Với sự hỗ trợ của ngôn ngữ Python và các thư viện liên quan, thuật toán khoảng cách Euclid được sử dụng để tiền xử lý và so sánh các đặc điểm khuôn mặt, từ đó đưa ra kết quả đối chiếu. Cuối cùng, bước ba là đẩy kết quả lên giao diện, nơi các giám thị có thể xác nhận và thông báo kết quả đối chiếu, đảm bảo tính chính xác và hiệu quả trong quá trình xác minh danh tính sinh viên.

|  |  |
| --- | --- |
| **Mục đích** | Đối chiếu khuôn mặt sinh viên với thẻ sinh viên thực tế để xác minh danh tính. |
| **Các bước thực hiện** | **Bước 1:** Chuẩn bị dữ liệu đầu vào, đảm bảo ảnh thẻ sinh viên và ảnh thực tế của sinh viên được cắt đúng khuôn mặt và chính xác.  **Bước 2:** Xây dựng chương trình đối chiếu khuôn mặt sử dụng thuật toán khoảng cách Euclid trong Python để so sánh và đưa ra kết quả.  **Bước 3:** Đẩy kết quả đối chiếu lên giao diện để giám thị có thể xác nhận và thông báo kết quả. |
| **Các bước**  **bổ sung** | * Sử dụng thư viện hỗ trợ Python như OpenCV và dlib để thực hiện các thao tác tiền xử lý ảnh trước khi đối chiếu khuôn mặt. * Tăng cường độ chính xác của thuật toán qua các bài huấn luyện và cải tiến mô hình. * Tăng cường độ chính xác của thuật toán qua các bài huấn luyện và cải tiến mô hình. |
| **Các ngoại lệ** | * Ảnh đầu vào bị mờ hoặc không rõ nét, ảnh thực tế không trùng khớp với ảnh trên thẻ. * Khuôn mặt trong ảnh thực tế không đủ rõ ràng hoặc bị che khuất. |
| **Tài liệu liên quan** | Python, OpenCV, dlib, thuật toán khoảng cách Euclid. |
| **Tham chiếu** | Mục 1.6.1.3: Chức năng đối chiếu khuôn mặt với thẻ sinh viên.  Mục 3.3.1.3: Chức năng đối chiếu khuôn mặt với thẻ sinh viên.. |

Bảng 2.6: Bảng chức năng đối chiếu khuôn mặt thực tế với ảnh thẻ sinh viên.

* + 1. Thiết kế quy trình nghiệp vụ

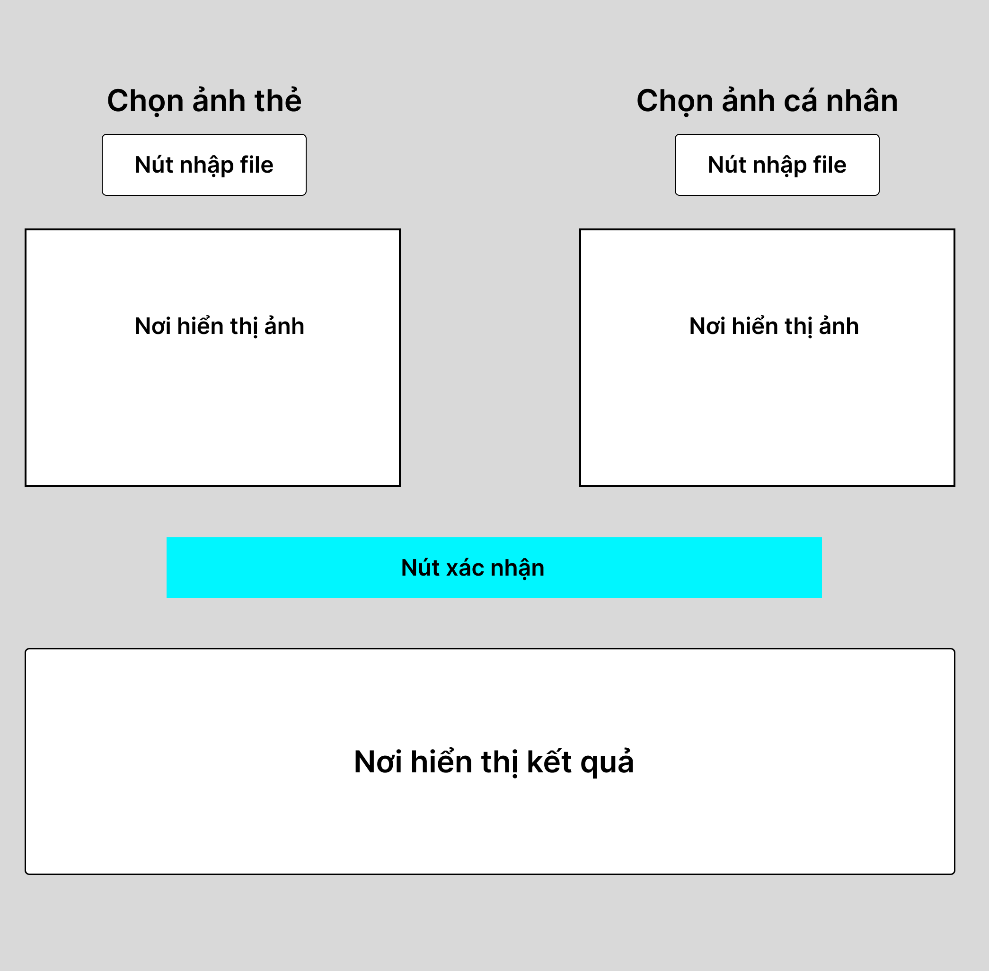
Quy trình đối chiếu khuôn mặt trong Sprint 3 bao gồm ba bước chính để đảm bảo tính chính xác và hiệu quả trong việc xác minh danh tính sinh viên. Đầu tiên, hệ thống sẽ trích xuất khuôn mặt từ cả ảnh thẻ sinh viên và ảnh thực tế, sau đó sử dụng các phương pháp Face Recognition để so sánh độ tương đồng giữa hai khuôn mặt. Nếu độ tương đồng đạt mức chấp nhận được, hệ thống sẽ xác nhận danh tính của sinh viên. Nếu không khớp, hệ thống sẽ yêu cầu kiểm tra lại. Tiếp theo, nếu khuôn mặt khớp, kết quả sẽ được lưu vào cơ sở dữ liệu và cập nhật để xác nhận danh tính. Ngược lại, hệ thống sẽ ghi lại kết quả và thông báo cho giám thị để kiểm tra lại thủ công. Cuối cùng, sau khi hoàn tất quá trình đối chiếu, hệ thống sẽ thông báo kết quả cho sinh viên và giám thị qua giao diện, giúp xác nhận thành công hoặc thất bại trong việc đối chiếu khuôn mặt.



Hình 2.8: Biểu đồ tuần tự chức năng đối chiếu khuôn mặt thực tế với ảnh thẻ sinh viên

* + 1. Thiết kế giao diện

**Hình 2.9** mô tả giao diện mẫu của chức năng đọc danh sách sinh viên trong hệ thống. Giao diện bao gồm hai mục chính: nơi tiếp nhận ảnh thẻ sinh viên và ảnh thẻ thực tế. Ảnh thẻ sinh viên được tải lên sẽ trải qua quá trình xử lý. Kết quả sau khi xử lý sẽ được hiển thị tại khu vực "Nơi hiển thị kết quả", giúp giám thị dễ dàng theo dõi và xác minh thông tin của sinh viên.



Hình 2.9: Mẫu giao diện chức năng đối chiếu khuôn mặt

* 1. Sprint 4: Xuất phiếu thi và giám sát quy trình thi
     1. Giới thiệu Sprint 4

Sprint 4 tập trung vào hai nhiệm vụ chính. Đầu tiên, hệ thống sẽ thực hiện việc xuất phiếu thi cá nhân cho sinh viên, sử dụng thông tin đã lưu trong cơ sở dữ liệu để tạo phiếu thi với đầy đủ thông tin cá nhân và lịch thi. Phiếu thi sẽ được xuất dưới dạng PDF. Nhiệm vụ thứ hai là phát triển Mô-đun giám sát quy trình thi, nhằm phát hiện các hành vi gian lận, xác minh danh tính sinh viên và theo dõi vị trí thi theo thời gian thực. Mô-đun này sẽ kết hợp công nghệ nhận diện khuôn mặt và giám sát từ camera, đảm bảo tính minh bạch và công bằng trong suốt kỳ thi.

* + 1. Thiết kế dữ liệu

Hiện tại, chúng tôi mới chỉ triển khai được chức năng in phiếu thi. Vì phiếu thi chỉ là công cụ tạm thời, không mang tính chất lưu trữ lâu dài và sẽ bị huỷ ngay sau khi ca thi hoàn thành, nên việc lưu trữ phiếu thi là không cần thiết. Do đó, phiếu thi sẽ gồm các trường thông tin cơ bản như: *tên sinh viên*, *mã sinh viên*, *môn thi*, *thời gian thi*, và *tài khoản*, *mật khẩu đăng nhập*. Các trường này sẽ được lấy trực tiếp hoặc tự sinh từ cơ sở dữ liệu của hệ thống. Phiếu thi được triển khai dưới dạng PDF, giúp dễ dàng in ra hoặc gửi qua email cho sinh viên trước khi kỳ thi diễn ra.

* + 1. Quy trình xuất phiếu thi

Quy trình xuất phiếu thi được thiết kế nhằm tự động hóa việc tạo phiếu thi cho sinh viên, từ việc truy xuất thông tin sinh viên trong cơ sở dữ liệu đến việc tạo và gửi phiếu thi cho sinh viên. Hệ thống sẽ sử dụng dữ liệu đã lưu trữ, bao gồm thông tin cá nhân và lịch thi, để tạo ra phiếu thi chuẩn xác. Phiếu thi sẽ được xuất dưới dạng PDF chứa thông tin quan trọng giúp giám thị và sinh viên xác minh dễ dàng tại phòng thi. Toàn bộ quy trình không chỉ giúp tiết kiệm thời gian mà còn đảm bảo tính chính xác và thuận tiện trong việc cấp phát phiếu thi cho sinh viên.

|  |  |
| --- | --- |
| **Mục đích** | Tạo phiếu thi cá nhân cho sinh viên |
| **Các bước thực hiện** | **Bước 1:** Truy xuất dữ liệu sinh viên từ cơ sở dữ liệu để lấy thông tin cá nhân và lịch thi.  **Bước 2:** Tạo phiếu thi bằng cách định dạng dữ liệu vào mẫu phiếu thi chuẩn, bao gồm các thông tin như mã sinh viên, tên, lớp, ngày thi, giờ thi và địa điểm thi, tự sinh dữ liệu tài khoản và mật khẩu ca thi.  **Bước 3:** Xuất phiếu thi dưới dạng file PDF phục vụ việc in phiếu để dễ dàng tiến hành tại phòng thi.  **Bước 4:** Sinh viên kiểm tra phiếu thi và xác nhận thông tin, yêu cầu chỉnh sửa nếu có sai sót. |
| **Các bước**  **bổ sung** | Sử dụng thư viện iText PDF và Java Spring Boot để tạo và xuất file PDF. |
| **Các ngoại lệ** | * Thiếu thông tin sinh viên trong cơ sở dữ liệu hoặc lỗi khi truy xuất dữ liệu. * Không thể tạo hoặc xuất thành công file PDF |
| **Tài liệu liên quan** | Java Spring Boot, iText PDF Library, MongoDB. |
| **Tham chiếu** | Mục 1.7.1.3: Chức năng xuất phiếu thi cá nhân cho sinh viên.  Mục 3.3.1.3: Chức năng xuất phiếu thi cá nhân cho sinh viên. |

Bảng 2.7: Bảng chức năng xuất phiếu thi cá nhân cho sinh viên

* + 1. Thiết kế quy trình nghiệp vụ

Quy trình xuất phiếu thi và giám sát quy trình thi bắt đầu bằng việc hệ thống nhận danh sách thí sinh và thông tin liên quan đến phòng thi như chỗ ngồi, etc. Dựa trên các dữ liệu này, hệ thống sẽ tự động tạo phiếu thi cho từng thí sinh, trong đó ghi rõ thông tin cá nhân, tên đăng nhập, mập khẩu, thời gian thi, và hướng dẫn liên quan. Các phiếu thi này sau đó được xuất ra ở định dạng PDF hoặc in trực tiếp để phân phát đến từng thí sinh trước khi thi diễn ra.

Trong suốt quá trình thi, hệ thống sẽ giám sát chặt chẽ bằng cách sử dụng các công cụ hỗ trợ, như nhận diện khuôn mặt để xác minh danh tính thí sinh khi vào phòng thi. Đồng thời, hệ thống ghi lại toàn bộ hành động của thí sinh qua camera hoặc log dữ liệu để đảm bảo không có hành vi gian lận xảy ra. Nếu phát hiện bất thường, hệ thống sẽ tự động cảnh báo cho giám thị để can thiệp kịp thời. Sau khi hoàn tất kỳ thi, kết quả giám sát và các dữ liệu liên quan sẽ được lưu trữ lại, phục vụ cho việc báo cáo và đánh giá sau này. Quy trình này không chỉ đảm bảo tính minh bạch mà còn tăng cường độ tin cậy trong quản lý thi trắc nghiệm.

A diagram with black text

Description automatically generated

Hình 2.10: Biểu đồ tuần tự xuất phiếu thi và giám sát quy trình thi

* 1. Sprint 5: Các chức năng quản lý của quản trị viên

Sprint 5 tập trung vào việc triển khai các chức năng quản lý dành cho quản trị viên, để dễ dàng theo dõi, quản lý và điều phối các hoạt động trong hệ thống. Hiện nay, hầu hết các cơ sở giáo dục đã áp dụng các hệ thống quản lý thông minh để tự động hóa và tối ưu hóa quy trình thi cử. Do đó, trong phạm vi đồ án này, chúng tôi chỉ xây dựng lại một cách sơ bộ các tính năng liên quan trực tiếp đến vận hành kỳ thi mà vẫn đảm bảo hệ thống hoạt động hiệu quả, nhằm tiết kiệm thời gian và kịp tiến độ.

Các chức năng chính sẽ bao gồm việc quản lý danh sách sinh viên, giám thị, phòng thi, và các kỳ thi. Hệ thống sẽ cung cấp giao diện quản trị để quản trị viên có thể dễ dàng thêm, sửa, xóa thông tin sinh viên và các thông tin liên quan đến kỳ thi. Quản trị viên cũng có thể truy cập vào các báo cáo thống kê liên quan đến việc tham gia thi của sinh viên, kết quả thi và các hoạt động giám sát trong suốt kỳ thi. Việc triển khai các chức năng này nhằm tăng cường khả năng giám sát và quản lý hệ thống một cách hiệu quả, đồng thời nâng cao tính bảo mật và minh bạch trong quy trình thi.

* + 1. Chức năng quản lý sinh viên
       1. Giới thiệu chức năng quản lý sinh viên của quản trị viên

Chức năng quản lý sinh viên của quản trị viên cho phép quản lý thông tin sinh viên như thêm mới, chỉnh sửa, xóa và tìm kiếm. Quản trị viên có thể cập nhật kết quả thi và xuất báo cáo về số lượng sinh viên, tình trạng tham gia thi và kết quả thi, giúp giám sát và quản lý hiệu quả trong quá trình tổ chức kỳ thi.

A diagram of a person

Description automatically generated

Hình 2.10: Biểu đồ ca sử dụng chức năng quản lý sinh viên

* + - 1. Thiết kế các bảng dữ liệu liên quan

Các bảng dữ liệu liên quan đến chức năng quản lý sinh viên chủ yếu kế thừa từ các bảng đã được xây dựng trước đó. Chức năng này tập trung vào việc thực hiện các câu lệnh truy vấn đơn giản để quản lý thông tin sinh viên, bao gồm việc tìm kiếm, cập nhật, và xóa dữ liệu. Các bảng dữ liệu sẽ bao gồm thông tin sinh viên, kết quả thi và lịch thi, giúp quản trị viên dễ dàng theo dõi và quản lý các hoạt động liên quan đến sinh viên trong quá trình thi. Bổ sung bảng kết quả thi của sinh viên có tên ExamResults.

* + - 1. Quy trình quản lý sinh viên

Trong quá trình quản lý sinh viên, các chức năng CRUD (Create, Read, Update, Delete) đóng vai trò quan trọng trong việc đảm bảo tính linh hoạt và chính xác của hệ thống. Cụ thể, chức năng thêm sinh viên (Create) cho phép quản trị viên nhập thông tin mới vào hệ thống; chức năng cập nhật (Update) giúp chỉnh sửa thông tin sinh viên khi có thay đổi; chức năng xóa (Delete) hỗ trợ việc loại bỏ dữ liệu không còn cần thiết; và chức năng tìm kiếm (Read) cho phép quản trị viên nhanh chóng tra cứu thông tin sinh viên. Các bước thực hiện cho mỗi chức năng như sau:

|  |  |
| --- | --- |
| **Mục đích** | **Các bước thực hiện** |
| Thêm mới sinh viên | 1. Quản trị viên nhập vào thông tin sinh viên.  2. Hệ thống thực hiện câu lệnh thêm vào cơ sở dữ liệu.  3. Hệ thống xác nhận việc thêm mới thành công và cập nhật lại danh sách sinh viên. |
| Cập nhật thông tin sinh viên | 1. Quản trị viên tìm kiếm sinh viên cần cập nhật thông tin.  2. Hệ thống truy vấn và hiển thị thông tin sinh viên.  3. Quản trị viên chỉnh sửa thông tin và gửi cập nhật.  4. Hệ thống thực hiện câu lệnh cập nhật vào cơ sở dữ liệu. |
| Xóa sinh viên | 1. Quản trị viên tìm kiếm và chọn sinh viên cần xóa.  2. Hệ thống xác nhận yêu cầu xóa sinh viên.  3. Quản trị viên xác nhận và hệ thống thực hiện câu lệnh xóa trong cơ sở dữ liệu. |
| Tìm kiếm sinh viên | 1. Quản trị viên nhập thông tin cần tìm kiếm (mã sinh viên, tên, lớp học,...) vào ô tìm kiếm.  2. Hệ thống thực hiện câu lệnh truy vấn để tìm kiếm thông tin sinh viên trong cơ sở dữ liệu.  3. Hệ thống hiển thị kết quả tìm kiếm cho quản trị viên. |

Bảng 2.8: Quy trình quản lý sinh viên của quản trị viên

A screenshot of a computer screen

Description automatically generated

Hình 2.11: Biều đồ tuần tự của chức năng quản lý sinh viên

* + - 1. A screenshot of a computer

         Description automatically generatedThiết kế giao diện quản lý sinh viên

Hình 2.12: Mẫu giao diện trang quản lý sinh viên

* + 1. Chức năng quản lý giám thị
       1. Giới thiệu chức năng quản lý giám thị của quản trị viên

Chức năng quản lý giám thị của quản trị viên giúp quản lý thông tin liên quan đến giám thị trong kỳ thi, bao gồm các chức năng như thêm mới, chỉnh sửa, xóa và tìm kiếm giám thị theo các tiêu chí như mã giám thị, tên và phòng thi. Quản trị viên có thể phân công giám thị cho các phòng thi, theo dõi lịch trình làm việc của giám thị và cập nhật thông tin về kết quả giám sát trong suốt kỳ thi. Điều này giúp đảm bảo tính minh bạch và công bằng trong việc giám sát kỳ thi, cũng như hỗ trợ quản lý hiệu quả việc phân công công việc cho giám thị.

A diagram of a person's process

Description automatically generated

Hình 2.13: Biểu đồ ca sử dụng chức năng quản lý giám thị

* + - 1. Thiết kế các bảng dữ liệu liên quan

Các bảng dữ liệu liên quan đến chức năng quản lý giám thị chủ yếu kế thừa từ các bảng đã được xây dựng trước đó. Chức năng này tập trung vào việc thực hiện các câu lệnh truy vấn đơn giản để quản lý thông tin sinh viên, bao gồm việc tìm kiếm, cập nhật, và xóa dữ liệu. Các bảng dữ liệu sẽ bao gồm thông tin sinh viên, kết quả thi và lịch thi, giúp quản trị viên dễ dàng theo dõi và quản lý các hoạt động liên quan đến sinh viên trong quá trình thi.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Diễn giải** | **Tên trường** | **Kiểu** | **Mặc định** | **Ràng buộc** | **Ghi chú** |
| 1 | SQLite tự tạo | proctor\_id | \_id |  |  | Đối tượng Sqlite tự sinh |
| 2 | Mã giám thị | proctor\_code | Text |  | PK | Khoá chính |
| 3 | Họ tên giám thị | full\_name | Text |  |  |  |
| 4 | Đơn vị | department | Text |  |  |  |
| 4 | Số điện thoại | phone\_number | Text |  |  |  |
| 5 | Email | email | Text |  |  |  |

Bảng 2.9: Bảng mô tả dữ liệu lưu trữ trong tbl Proctor

* + - 1. Quy trình quản lý giám thị

Quy trình quản lý giám thị đóng vai trò quan trọng trong việc duy trì sự hiệu quả và chính xác của hệ thống, với sự hỗ trợ của các chức năng CRUD (Tạo, Đọc, Cập nhật, Xóa). Chức năng **thêm giám thị** cho phép quản trị viên nhập thông tin giám thị mới, bao gồm các trường dữ liệu như họ tên, mã số, thông tin liên hệ và các thông tin liên quan, đảm bảo danh sách giám thị luôn được cập nhật đầy đủ. Chức năng **cập nhật thông tin** giúp chỉnh sửa hoặc bổ sung thông tin khi có thay đổi, ví dụ như thay đổi số điện thoại, địa chỉ hoặc chức vụ, để đảm bảo tính chính xác và đồng bộ trong dữ liệu.

Ngoài ra, chức năng **xóa giám thị** hỗ trợ loại bỏ những dữ liệu không còn cần thiết, chẳng hạn như thông tin của giám thị đã nghỉ việc hoặc không còn tham gia quản lý kỳ thi, giúp hệ thống gọn gàng và tối ưu hóa quản lý. Chức năng **tìm kiếm** là công cụ không thể thiếu, cho phép quản trị viên tra cứu nhanh chóng thông tin cần thiết dựa trên các tiêu chí như họ tên, mã giám thị, hoặc chức vụ, từ đó tiết kiệm thời gian và nâng cao hiệu suất công việc.

Những chức năng này được thiết kế và thực hiện theo các bước rõ ràng và nhất quán, đảm bảo rằng quy trình quản lý giám thị diễn ra một cách mạch lạc, chính xác và đáp ứng được nhu cầu vận hành của hệ thống trong dài hạn.

|  |  |
| --- | --- |
| **Mục đích** | **Các bước thực hiện** |
| Thêm mới giám thị | 1. Quản trị viên nhập thông tin vào giao diện quản lý.  2. Hệ thống thực hiện câu lệnh thêm vào cơ sở dữ liệu.  3. Hệ thống xác nhận việc thêm mới thành công và cập nhật lại danh sách sinh viên. |
| Cập nhật thông tin giám thị | 1. Quản trị viên tìm kiếm giám thị cần cập nhật thông tin.  2. Hệ thống truy vấn và hiển thị thông tin đã chọn.  3. Quản trị viên chỉnh sửa thông tin cần thay đổi.  4. Hệ thống thực hiện câu lệnh cập nhật vào cơ sở dữ liệu. |
| Xóa giám thị | 1. Quản trị viên tìm kiếm và chọn giám thị cần xóa.  2. Hệ thống xác nhận yêu cầu xóa giám thị.  3. Quản trị viên xác nhận lệnh xóa trong cơ sở dữ liệu. |
| Tìm kiếm giám thị | 1. Quản trị viên nhập thông tin cần tìm kiếm (mã giám thị, họ tên, đơn vị,…) vào ô tìm kiếm.  2. Hệ thống thực hiện câu lệnh truy vấn để tìm kiếm thông tin giám thị trong cơ sở dữ liệu.  3. Hệ thống hiển thị kết quả tìm kiếm cho quản trị viên. |

Bảng 2.10: Quy trình quản lý giám thị của quản trị viên

A screenshot of a diagram

Description automatically generated

Hinh 2.14: Biểu đồ tuần tự của chức năng quản lý giám thị

* + - 1. Thiết kế giao diện quản lý giám thị

A screenshot of a computer

Description automatically generated

Hình 2.15: Mẫu giao diện chức năng quản lý giám thị

A screenshot of a login form

Description automatically generated

Hình 2.16: Mẫu giao diện chức năng thêm mới giám thị

* + 1. Chức năng quản lý phòng thi
       1. Giới thiệu chức năng quản lý phòng thi của quản trị viên

Chức năng quản lý phòng thi của quản trị viên giúp quản lý thông tin liên quan đến phòng thi trong kỳ thi, bao gồm các chức năng như thêm mới, chỉnh sửa, xóa và tìm kiếm phòng thi theo các tiêu chí như mã phòng, địa điểm và số lượng thí sinh. Quản trị viên có thể linh hoạt phân bổ thí sinh và giám thị vào từng phòng thi, theo dõi tình trạng sử dụng phòng thi, cũng như cập nhật thông tin về kết quả thi trong mỗi phòng. Chức năng này giúp đảm bảo tổ chức kỳ thi một cách hiệu quả, minh bạch và thuận tiện trong việc giám sát, đồng thời tối ưu hóa việc sử dụng tài nguyên phòng thi.

A diagram of a diagram

Description automatically generated

Hình 2.17: Biểu đồ ca sử dụng chức năng quản lý phòng thi

* + - 1. Thiết kế các bảng dữ liệu liên quan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Diễn giải** | **Tên trường** | **Kiểu** | **Mặc định** | **Ràng buộc** | **Ghi chú** |
| 1 | Mã phòng thi | room\_code | Text |  | PK | Khoá chính |
| 2 | Địa chỉ phòng thi | address | Text |  |  |  |
| 3 | Số lượng sinh viên tối đa | max\_capacity | Text |  |  |  |
| 4 | Danh sách camera | cameras | Text(camera\_id) |  |  |  |
| 5 | Danh sách máy tính | computers | Text(computer\_id) |  |  |  |

Bảng 2.11: Bảng mô tả dữ liệu lưu trữ trong tbl ExamRooms

* + - 1. Quy trình quản lý phòng thi

|  |  |
| --- | --- |
| **Mục đích** | **Các bước thực hiện** |
| Thêm mới phòng thi | 1. Quản trị viên nhập thông tin vào giao diện quản lý.  2. Hệ thống thực hiện câu lệnh thêm vào cơ sở dữ liệu.  3. Hệ thống xác nhận việc thêm mới thành công và cập nhật lại danh sách phòng thi. |
| Cập nhật thông tin phòng thi | 1. Quản trị viên tìm kiếm phòng thi cần cập nhật thông tin.  2. Hệ thống truy vấn và hiển thị thông tin đã chọn.  3. Quản trị viên chỉnh sửa thông tin cần thay đổi và gửi yêu cầu cập nhật.  4. Hệ thống thực hiện câu lệnh cập nhật vào cơ sở dữ liệu. |
| Xóa phòng thi | 1. Quản trị viên tìm kiếm và chọn phòng thi cần xóa.  2. Hệ thống xác nhận yêu cầu xóa phòng thi.  3. Quản trị viên xác nhận và hệ thống thực hiện câu lệnh xóa trong cơ sở dữ liệu. |

Bảng 2.12: Quy trình quản lý phòng thi của quản trị viên

* + - 1. Thiết kế giao diện quản lý phòng thi

A screenshot of a computer

Description automatically generated

Hình 2.18: Mẫu giao diện chức năng quản lý phòng thi

* 1. Biểu đồ ca sử dụng

Biểu đồ ca sử dụng trong hệ thống quản lý kỳ thi giúp mô tả các chức năng mà hệ thống cung cấp và cách thức các người dùng tương tác với hệ thống. Biểu đồ này cung cấp cái nhìn tổng quan về các yêu cầu chức năng của hệ thống và mô tả các tác vụ mà mỗi đối tượng (người dùng là quản trị viên, giám thị, sinh viên) có thể thực hiện.

A diagram of a diagram

Description automatically generated

Hình 2.18: Biểu đồ ca sử dụng hệ thống

1. XÂY DỰNG HỆ THỐNG VẬN HÀNH VÀ QUẢN LÝ QUY TRÌNH THI TRẮC NGHIỆM KHÁCH QUAN
   1. Môi trường phát triển
      1. Giới thiệu công nghệ và công cụ sử dụng
         1. Java Spring Boot

Java Spring Boot là một framework mạnh mẽ và phổ biến, được sử dụng rộng rãi trong phát triển các ứng dụng web và API backend. Được xây dựng trên nền tảng của Spring Framework, Spring Boot mang đến nhiều tính năng tự động hóa và tối ưu hóa quá trình phát triển, giúp giảm thiểu công việc cấu hình thủ công và nâng cao hiệu suất phát triển ứng dụng. Nhờ vào các tính năng này, Spring Boot trở thành một công cụ không thể thiếu cho các dự án lớn, đặc biệt trong việc xây dựng các hệ thống cần sự linh hoạt, mở rộng và bảo mật cao.

Một trong những ưu điểm nổi bật của Spring Boot là khả năng cấu hình tự động, giúp giảm thiểu tối đa công việc cấu hình ban đầu. Việc tạo ra một dự án Spring Boot chỉ yêu cầu một vài bước đơn giản, và framework này sẽ tự động cấu hình các phần mềm cần thiết cho việc triển khai ứng dụng. Điều này giúp giảm bớt gánh nặng cho các lập trình viên và giúp họ tập trung vào việc phát triển tính năng của ứng dụng thay vì lo lắng về cấu hình phức tạp.

Spring Boot còn hỗ trợ mạnh mẽ trong việc xây dựng các RESTful API và tích hợp với các cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, và MongoDB. Điều này đặc biệt hữu ích khi xây dựng các ứng dụng cần xử lý và lưu trữ lượng lớn dữ liệu, chẳng hạn như hệ thống quản lý quy trình thi trắc nghiệm, nơi cần lưu trữ thông tin sinh viên, giám thị, phòng thi, và kết quả thi.

Bên cạnh đó, Spring Boot cũng cung cấp các tính năng bảo mật mạnh mẽ thông qua Spring Security, cho phép quản lý quyền truy cập người dùng và bảo vệ các tài nguyên trong hệ thống. Điều này đảm bảo rằng chỉ những người có quyền mới có thể truy cập vào các chức năng quan trọng như tạo ca thi, xác minh thông tin sinh viên, hoặc quản lý giám thị.

Với khả năng mở rộng linh hoạt, hiệu suất cao và sự hỗ trợ tốt từ cộng đồng, Spring Boot là lựa chọn lý tưởng để phát triển backend cho hệ thống quản lý quy trình thi trắc nghiệm. Nó giúp đảm bảo hệ thống hoạt động ổn định, bảo mật và có thể mở rộng khi cần thiết để đáp ứng nhu cầu phát triển trong tương lai.

* + - 1. Flask Python

Flask là một microframework nhẹ nhàng và dễ sử dụng cho việc phát triển các ứng dụng web và API bằng ngôn ngữ lập trình Python. Được thiết kế với triết lý "Keep it simple, stupid" (KISS), Flask mang đến sự linh hoạt cao, cho phép người phát triển dễ dàng mở rộng ứng dụng mà không bị gò bó bởi những quy tắc và cấu trúc phức tạp. Flask phù hợp với các dự án từ nhỏ đến lớn, đặc biệt là các ứng dụng yêu cầu tốc độ phát triển nhanh và khả năng mở rộng dễ dàng.

Một trong những điểm mạnh của Flask là tính đơn giản trong việc tạo và triển khai các ứng dụng web. Không giống như các framework đầy đủ như Django, Flask không yêu cầu người phát triển phải tuân theo các quy tắc nghiêm ngặt về cấu trúc dự án, giúp giảm thiểu sự phức tạp khi triển khai các ứng dụng đơn giản hoặc các mô-đun nhỏ trong các hệ thống phức tạp hơn. Điều này khiến Flask trở thành sự lựa chọn tuyệt vời khi cần phát triển các ứng dụng nhỏ gọn, như các dịch vụ microservice hoặc các API RESTful.

Flask hỗ trợ tốt việc tích hợp với nhiều công nghệ và công cụ khác nhau, đặc biệt là trong việc xây dựng các API RESTful. Với Flask, người phát triển có thể dễ dàng triển khai các dịch vụ web và giao tiếp giữa các hệ thống qua HTTP, JSON hoặc XML. Flask cũng hỗ trợ nhiều thư viện mở rộng giúp tăng cường tính năng của ứng dụng như Flask-SQLAlchemy (quản lý cơ sở dữ liệu), Flask-Security (bảo mật), và Flask-Login (quản lý phiên đăng nhập).

Trong dự án quản lý quy trình thi trắc nghiệm, Flask sẽ được sử dụng chủ yếu để triển khai mô-đun nhận diện khuôn mặt sinh viên. Flask có thể tích hợp với các thư viện xử lý ảnh và nhận diện khuôn mặt như OpenCV và dlib, giúp hệ thống xác minh danh tính sinh viên một cách nhanh chóng và chính xác. Khi sinh viên tham gia thi, hệ thống sẽ đối chiếu khuôn mặt thực tế của sinh viên với ảnh đã được lưu trữ trong cơ sở dữ liệu để đảm bảo tính chính xác của thông tin và ngăn chặn gian lận.

Flask cũng nổi bật với khả năng tích hợp dễ dàng với các ứng dụng khác và hệ thống microservices. Điều này giúp việc xây dựng các ứng dụng có thể mở rộng và duy trì lâu dài trở nên thuận lợi hơn, đồng thời giúp tiết kiệm thời gian và chi phí trong việc phát triển các tính năng mới hoặc thay đổi yêu cầu hệ thống.

Nhờ vào sự linh hoạt, dễ dàng mở rộng và khả năng xử lý nhanh chóng, Flask là công cụ lý tưởng cho việc phát triển các mô-đun nhận dạng khuôn mặt trong hệ thống quản lý thi trắc nghiệm, giúp hệ thống hoạt động nhanh chóng, chính xác và hiệu quả.

* + - 1. Hệ quản trị cơ sở dữ liệu SQLite

SQLite là một hệ quản trị cơ sở dữ liệu quan hệ nhẹ, nhúng (embedded) và mã nguồn mở, được thiết kế để dễ sử dụng và tích hợp trực tiếp vào ứng dụng mà không yêu cầu cài đặt và cấu hình phức tạp. SQLite là một lựa chọn phổ biến cho các ứng dụng yêu cầu cơ sở dữ liệu nhẹ, nhanh chóng và dễ dàng triển khai, đặc biệt là khi không cần một hệ quản trị cơ sở dữ liệu phức tạp như MySQL hoặc PostgreSQL. Nó phù hợp với các dự án có kích thước vừa và nhỏ, hoặc các ứng dụng cần đơn giản hóa việc quản lý dữ liệu mà không làm giảm hiệu suất.

SQLite lưu trữ dữ liệu trong một tệp duy nhất, điều này giúp đơn giản hóa việc triển khai và sao lưu dữ liệu. Vì không cần một máy chủ riêng biệt để vận hành, SQLite rất dễ tích hợp vào các ứng dụng, không yêu cầu quá nhiều tài nguyên hệ thống, và có thể chạy trên hầu hết các nền tảng, từ máy tính cá nhân đến thiết bị di động.

Trong dự án quản lý quy trình thi trắc nghiệm, SQLite sẽ được sử dụng làm hệ quản trị cơ sở dữ liệu chính để lưu trữ thông tin sinh viên, giám thị, các phòng thi, các ca thi và kết quả thi. Cơ sở dữ liệu sẽ bao gồm nhiều bảng, mỗi bảng lưu trữ các thông tin quan trọng, chẳng hạn như bảng Students (lưu thông tin sinh viên), Proctors (lưu thông tin giám thị), và ExamRooms (lưu thông tin phòng thi). Các bảng này sẽ liên kết với nhau thông qua các khóa ngoại, tạo thành một hệ thống dữ liệu hoàn chỉnh và dễ dàng truy vấn.

SQLite có một số đặc điểm nổi bật trong việc sử dụng cho dự án này:

**Nhẹ và dễ triển khai**: SQLite không yêu cầu cài đặt máy chủ cơ sở dữ liệu riêng biệt, điều này giúp đơn giản hóa việc triển khai hệ thống, đặc biệt khi cần phát triển ứng dụng nhanh chóng.

**Tính linh hoạt**: SQLite hỗ trợ nhiều kiểu dữ liệu và các câu lệnh SQL chuẩn, giúp dễ dàng quản lý và truy xuất thông tin từ cơ sở dữ liệu.

**Hiệu suất cao**: SQLite hoạt động nhanh và hiệu quả trong các ứng dụng có yêu cầu truy xuất dữ liệu đơn giản và có số lượng bản ghi vừa phải, như trong trường hợp quản lý danh sách sinh viên và phòng thi.

**Khả năng tương thích cao**: SQLite có thể hoạt động trên nhiều nền tảng khác nhau, từ máy tính để bàn, máy chủ, đến thiết bị di động, giúp hệ thống có thể được triển khai trên nhiều môi trường.

SQLite sẽ được sử dụng để lưu trữ thông tin về sinh viên (bao gồm mã sinh viên, tên, ngành học, khóa học, và ảnh thẻ), giám thị (mã giám thị, tên, phòng thi phân công), các ca thi (thời gian thi, phòng thi, danh sách sinh viên tham gia), và kết quả thi của sinh viên. Ngoài ra, SQLite cũng sẽ lưu trữ các bản ghi lịch sử và thông tin về trạng thái thi, như các sinh viên có mặt hay vắng mặt, giúp giám thị và admin dễ dàng theo dõi quá trình thi.

Nhờ vào tính nhẹ nhàng và hiệu quả, SQLite là sự lựa chọn hoàn hảo cho dự án quản lý thi trắc nghiệm, giúp tiết kiệm tài nguyên và mang lại sự linh hoạt trong việc quản lý dữ liệu, đồng thời đảm bảo hệ thống hoạt động ổn định và dễ bảo trì.

* + 1. Quản lý mã nguồn bằng Github

GitHub là một nền tảng quản lý mã nguồn và hợp tác phát triển dựa trên hệ thống kiểm soát phiên bản Git. Việc sử dụng GitHub trong đề tài giúp nhóm phát triển theo dõi sự thay đổi của mã nguồn, quản lý các phiên bản và hợp tác hiệu quả hơn khi làm việc nhóm.

GitHub hỗ trợ việc phân nhánh (branching) và hợp nhất (merging) dễ dàng, giúp phát triển và thử nghiệm các tính năng mới mà không làm gián đoạn phiên bản chính của hệ thống. Ngoài ra, GitHub cung cấp công cụ để quản lý vấn đề, theo dõi tiến độ, và tích hợp CI/CD để tự động hoá việc kiểm thử và triển khai ứng dụng, đảm bảo quy trình phát triển liên tục và linh hoạt.

* + 1. Phân chia công việc, quản lý tiến độ đề tài bằng Trello

Trello là một công cụ quản lý đề tài trực tuyến dựa trên phương pháp Kanban, giúp theo dõi tiến độ công việc một cách trực quan và hiệu quả. Trong đề tài này, Trello được sử dụng để phân chia công việc giữa các thành viên trong nhóm, từ đó đảm bảo mọi người có thể theo dõi tiến độ công việc và điều chỉnh kế hoạch khi cần thiết.

Mỗi công việc sẽ được đại diện dưới dạng các thẻ trong các cột như "To Do", "In Progress" và "Completed", giúp dễ dàng xác định tình trạng hiện tại của từng nhiệm vụ. Các nhiệm vụ được phân chia theo các giai đoạn của đề tài, ví dụ như phân tích yêu cầu, phát triển tính năng, kiểm thử và triển khai.

Trello cũng cho phép thêm các deadline, bình luận, đính kèm tài liệu và phân công nhiệm vụ cho các thành viên cụ thể, tạo nên một không gian làm việc hợp tác và có tổ chức. Bằng cách này, Trello giúp nhóm theo dõi tiến độ đề tài, đảm bảo đúng thời hạn và tối ưu hóa hiệu quả công việc.

* 1. Chức năng cập nhật ảnh cá nhân của sinh viên lên hệ thống

Chức năng cập nhật ảnh cá nhân cho phép sinh viên tải lên và thay đổi ảnh thẻ sinh viên của mình trong hệ thống quản lý thi trắc nghiệm. Sinh viên có thể đăng nhập vào tài khoản của mình, truy cập vào phần thông tin cá nhân và chọn ảnh mới từ thiết bị của mình để tải lên. Hệ thống sẽ lưu trữ ảnh đã cập nhật vào cơ sở dữ liệu và liên kết với các thông tin cá nhân của sinh viên, đảm bảo quá trình xác minh danh tính khi thi được thực hiện chính xác và nhanh chóng. Chức năng này giúp duy trì thông tin sinh viên luôn được cập nhật, góp phần nâng cao tính chính xác trong việc xác thực và đối chiếu danh tính sinh viên trong phòng thi.

* 1. Xây dựng các thành phần liên quan
     + 1. Students Entity

Students Entity là một lớp trong hệ thống quản lý thi trắc nghiệm, đại diện cho bảng Students trong cơ sở dữ liệu. Lớp này chứa các thuộc tính và phương thức cần thiết để lưu trữ và truy xuất thông tin của sinh viên, bao gồm các trường thông tin cá nhân và học thuật của sinh viên. Mỗi đối tượng của Students Entity sẽ tương ứng với một bản ghi trong bảng Students, giúp hệ thống quản lý dữ liệu sinh viên dễ dàng.

@Entity  
@Table(name="Students")  
public class Student {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long studentId;

@Column(nullable = false, unique = true)  
 private String studentCode;

@Column(nullable = false)  
 private String fullName;

@Column(nullable = false)  
 private String department;

@Column(nullable = false)  
 private String className;

@Column(nullable = false)  
 private String cohort;

@Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private Gender gender;

@Convert(converter = LocalDateConverter.class)  
 @Column(nullable = false)  
 private LocalDate dateOfBirth;

@Column(nullable = true)  
 private String phoneNumber;  
  
 @Column(nullable = true)  
 private String email;

@Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private Status status;

@Column(nullable = true)  
 private String studentPhoto;

@Column(nullable = true)  
 private String facePhoto;

Hình 3.1: Ảnh mã nguồn Student Entity

* + - 1. Student Repository

Student Repository là một lớp trong hệ thống quản lý thi trắc nghiệm, chịu trách nhiệm thực hiện các thao tác truy xuất và quản lý dữ liệu liên quan đến sinh viên trong cơ sở dữ liệu. Lớp này sẽ sử dụng các phương thức của Spring Data JPA (hoặc tương tự) để tương tác với cơ sở dữ liệu, thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên bảng Students.

public interface StudentRepository extends JpaRepository<Student, Long> {  
 Optional<Student> findByStudentCode(String studentCode);  
  
 List<Student> findByFullNameContainingIgnoreCase(String fullName);  
  
 List<Student> findByClassName(String className);  
  
 List<Student> findByStatus(Student.Status status);  
}

Hình 3.2: Ảnh mã nguồn Student Repository

* + - 1. Student Service

Student Service là lớp chịu trách nhiệm xử lý các logic nghiệp vụ liên quan đến sinh viên trong hệ thống quản lý thi trắc nghiệm. Lớp này sẽ gọi đến Student Repository để thực hiện các thao tác truy xuất và cập nhật dữ liệu sinh viên, đồng thời cung cấp các dịch vụ cho các lớp khác trong ứng dụng, như lớp điều khiển (controller) hoặc các lớp dịch vụ khác. Student Service giúp tách biệt logic nghiệp vụ khỏi lớp điều khiển và lớp dữ liệu, đảm bảo tính mô-đun và khả năng bảo trì của ứng dụng.

@Service  
public class StudentService {  
  
 @Autowired  
 private StudentRepository studentRepository;  
  
 public List<Student> findAllStudents() {  
 return studentRepository.findAll();  
 }  
  
 public Student findStudentByStudentCode(String studentCode) {  
 return studentRepository.findByStudentCode(studentCode).orElse(null);  
 }  
  
 public List<Student> findStudentsByStudentCode(String studentCode) {  
 return studentRepository.findByStudentCode(studentCode)  
 .map(Collections::*singletonList*)  
 .orElse(Collections.*emptyList*());  
 }  
  
 public List<Student> findByFullName(String fullName) {  
 return studentRepository.findByFullNameContainingIgnoreCase(fullName);  
 }  
  
 public List<Student> findByClassName(String className) {  
 return studentRepository.findByClassName(className);  
 }  
  
 public List<Student> findByStatus(Student.Status status) {  
 return studentRepository.findByStatus(status);  
 }  
  
 public void save(Student student) {  
 studentRepository.save(student);  
 }  
  
 public void updateStudent(Student student) {  
 studentRepository.save(student);  
 }  
}

Hình 3.3: Ảnh mã nguồn Student Service

* + - 1. Student Controller

Student Controller là lớp điều khiển (controller) trong hệ thống quản lý thi trắc nghiệm, chịu trách nhiệm xử lý các yêu cầu từ phía người dùng (ví dụ: sinh viên, giám thị, quản trị viên) và điều hướng các yêu cầu này đến các dịch vụ thích hợp. Lớp này nhận các yêu cầu HTTP, gọi đến các phương thức trong Student Service để xử lý nghiệp vụ và trả về kết quả dưới dạng phản hồi HTTP.

@Controller  
@RequestMapping("/student")  
public class StudentController {  
  
 @Autowired  
 private UserService userService;  
  
 @Autowired  
 private StudentService studentService;  
  
 @Autowired  
 private ExamRoomService examRoomService;  
  
 @Autowired  
 private StudentRepository studentRepository;  
  
 @Autowired  
 private ExamRoomRepository examRoomRepository;  
  
 @Autowired  
 private ExamSessionRepository examSessionRepository;  
  
 // Trang dashboard-student  
 @GetMapping("/dashboard-student")  
 public String studentDashboard(Model model) {  
 String userCode = userService.getUserCode();  
// System.out.println("User Code: " + userCode);  
 if (userCode != null) {  
 Student student = studentService.findStudentByStudentCode(userCode);  
// System.out.println("Student Info: " + student);  
 if (student != null) {  
 model.addAttribute("student", student);  
 model.addAttribute("pageContent", "dashboard-student");  
 return "template-student";  
 } else {  
 model.addAttribute("error", "Student not found");  
 return "redirect:error";  
 }  
 } else {  
 model.addAttribute("error", "User code not found");  
 return "redirect:error";  
 }  
 }  
  
 @GetMapping("/info")  
 public String studentInfo(Model model) {  
 String userCode = userService.getUserCode();  
  
 if (userCode != null) {  
 Student student = studentService.findStudentByStudentCode(userCode);  
  
 if (student != null) {  
 if (student.getStudentPhoto() == null || student.getStudentPhoto().isEmpty()) {  
 student.setStudentPhoto("/images/default-student-photo.jpg");  
 }  
 if (student.getFacePhoto() == null || student.getFacePhoto().isEmpty()) {  
 student.setFacePhoto("/images/default-face-photo.jpg");  
 }  
  
 model.addAttribute("student", student);  
 model.addAttribute("pageContent", "student-info");  
 return "template-student";  
 } else {  
 model.addAttribute("error", "Student not found");  
 return "error";  
 }  
 } else {  
 model.addAttribute("error", "User code not found");  
 return "error";  
 }  
 }  
  
 @GetMapping("/schedule")  
 public String viewExamSchedule(Model model) {  
 String studentCode = userService.getUserCode();  
 if (studentCode == null) {  
 model.addAttribute("error", "Student code is required");  
 return "error";  
 }  
 Optional<Student> student = studentRepository.findByStudentCode(studentCode);  
 if (student.isEmpty()) {  
 model.addAttribute("error", "Sinh viên không tồn tại");  
 return "error";  
 }  
 List<ExamSession> examSessions = examSessionRepository.findByStudentCode(studentCode);  
  
 model.addAttribute("student", student.get());  
 model.addAttribute("examSessions", examSessions);  
  
 model.addAttribute("pageContent", "exam-schedule-std");  
 return "template-student";  
 }

Hình 3.4: Ảnh mã nguồn Student Controller

* + 1. Giao diện chức năng xử lý ảnh đầu vào và trích xuất thông tin trên thẻ

**Hình 3.5** mô tả giao diện chức năng trích xuất thông tin trên thẻ sinh viên, bao gồm ba khu vực chính. Đầu tiên là *khu vực tải ảnh*, nơi người dùng có thể dễ dàng tải ảnh lên thông qua nút *Tải ảnh*. Sau khi ảnh được tải lên, hệ thống sẽ hiển thị ảnh đầu vào trực tiếp trên giao diện, giúp người dùng kiểm tra và xác nhận ảnh cần xử lý. Tiếp theo là *khu vực kết quả nhận diện*, nơi hiển thị các thông tin đã trích xuất từ ảnh như tên sinh viên, mã số thẻ, ngày sinh, và các trường thông tin liên quan. Đồng thời, khu vực này còn có mục *Thông báo trạng thái* để cung cấp phản hồi về quá trình nhận diện, chẳng hạn như "*Nhận diện thành công*" hoặc "*Không nhận diện được thông tin*". Cuối cùng, các nút chức năng được bố trí để hỗ trợ người dùng thao tác. Sau khi tải ảnh, người dùng có thể nhấn nút *Xử lý ảnh* để hệ thống thực hiện quá trình nhận diện và trích xuất thông tin từ ảnh một cách tự động. Giao diện được thiết kế đơn giản và trực quan, đảm bảo sự tiện lợi và hiệu quả cho người sử dụng.

A screenshot of a computer

Description automatically generated

Hình 3.5: Giao diện chức năng cập nhật thông tin sinh viên

* + 1. Kiểm thử chức năng

Dưới đây là bảng trình bày các thử nghiệm chi tiết cho chức năng xử lý ảnh đầu vào và trích xuất thông tin trên thẻ sinh viên. Các ca thử nghiệm này bao gồm mục tiêu kiểm thử, kịch bản kiểm thử, kết quả mong đợi và kết quả thực tế để đảm bảo hệ thống hoạt động đúng như mong đợi trong các tình huống khác nhau.

Bảng dưới đây sẽ giúp bạn kiểm tra từng chức năng của hệ thống một cách cụ thể và có cơ sở để so sánh kết quả thực tế với kết quả mong đợi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mục tiêu kiểm thử** | **Kịch bản kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** |
| 1 | Kiểm thử tải ảnh lên | Nhấn nút “Tải ảnh” và chọn ảnh hợp lệ từ máy | Ảnh được tải lên thành công và hiển thị đúng trên giao diện | Như kết quả mong đợi |
| 2 | Kiểm thử tải ảnh với định dạng không hợp lệ | Chọn file có định dạng không phải là ảnh (ví dụ: .txt, .docx) để tải lên | Hệ thống thông báo lỗi về định dạng | Thông báo định dạng không hợp lệ |
| 3 | Kiểm thử xử lý ảnh với chất lượng cao | Sau khi tải ảnh lên. Kiểm tra kết quả trích xuất được từ ảnh rõ nét | Thông tin (họ tên sinh viên, mã số sinh viên, khoá học, ngành học,…) được trích xuất chính xác | Các thông tin được trích xuất đúng với thực tế |
| 4 | Kiểm thử xử lý với ảnh chất lượng kém | Tải ảnh bị mờ, nhiễu lên. Kiểm tra kết quả trích xuất | Hệ thống thông báo lỗi không trích xuất được thông tin | Trả về kết quả sai hoặc chưa đầy đủ. Không thông báo lỗi |
| 5 | Kiểm thử thông báo lỗi khi không thể nhận diện ảnh | Sử dụng ảnh không phải là thẻ sinh viên (ví dụ: ảnh phong cảnh) và tải ảnh lên | Hệ thống hiển thị thông báo lỗi “Không nhận diện được thông tin” | Thông báo lỗi như mong đợi |
| 6 | Kiểm thử với ảnh thẻ có định dạng khác | Sử dụng ảnh của khoá 61 (định dạng thẻ dọc), tải lên và xử lý | Hệ thống thông báo lỗi không đúng định dạng thẻ | Chưa thông báo lỗi |
| 7 | Kiểm thử giao diện người dùng (UI) | Kiểm tra tính năng các nút “Tải ảnh”, “Xử lý” trên giao diện người dùng | Giao diện dễ sử dụng, các nút chức năng hoạt động đúng và không gặp lỗi. | Giao diện dễ sử dụng, không gặp lỗi |
| 8 | Kiểm thử hệ thống phản hồi khi không có ảnh tải lên | Nhấn "Xử lý ảnh" khi chưa tải ảnh nào lên. | Hệ thống hiển thị thông báo lỗi "Chưa tải ảnh lên". | Thông báo lỗi "Chưa tải ảnh lên". |

Bảng 3.1: Kiểm thử chức năng xử lý ảnh đầu vào và trích xuất thông tin

* 1. Chức năng so khớp thông tin sinh viên với danh sách phòng thi

Chức năng này nhằm mục đích so khớp thông tin sinh viên đã đọc được ở bước xử lý ảnh đầu vào và trích xuất thông tin với danh sách thực tế có trong phòng thi, đảm bảo rằng mỗi sinh viên có thể được phân bổ và có mặt đúng phòng thi giúp việc quản lý và giám sát dễ dàng trong quá trình thi. Dưới đây là các Mô-đun cần thiết để thực hiện chức năng này.

* + 1. Xây dựng các Mô-đun liên quan
       1. Mô-đun đọc lấy danh sách từ cơ sở dữ liệu

**Mục đích**: Hàm này đọc dữ liệu từ file Excel chứa danh sách sinh viên, xác định các cột "Họ tên" và "Mã sinh viên", sau đó trả về danh sách sinh viên với định dạng "HỌ TÊN - MÃ SINH VIÊN".

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả chi tiết** |
| Tham số đầu vào | |  | | --- | |  |  |  | | --- | | excel\_path (str): Đường dẫn đến file Excel cần đọc. | |
| Thư viện sử dụng | pandas: Đọc và xử lý dữ liệu từ file Excel.  unidecode: Loại bỏ dấu tiếng Việt và chuẩn hóa chuỗi. |
| Quy trình hoạt động | 1. Đọc file Excel vào DataFrame bằng thư viện pandas.  2. Xác định cột "Họ tên" và "Mã sinh viên" dựa trên tiêu đề cột.  3. Kiểm tra giá trị hợp lệ, loại bỏ các ô dữ liệu trống hoặc không xác định.  4. Chuyển họ tên thành chữ in hoa, loại bỏ dấu và định dạng đầu ra theo yêu cầu.  5. Trả về danh sách chứa thông tin sinh viên đã được xử lý hoặc danh sách rỗng nếu gặp lỗi. |
| Kết quả trả về | list: Danh sách sinh viên dưới dạng ["HỌ TÊN - MÃ SINH VIÊN"]. |

Bảng 3.2: Bảng mô tả Mô-đun lấy danh sách từ cơ sở dữ liệu

def get\_student\_list(*session\_code*):

    conn = connect\_db()

    if not conn:

        print("Không thể kết nối đến cơ sở dữ liệu.")

        return {"error": "Không thể kết nối đến cơ sở dữ liệu."}

    try:

        cursor = conn.cursor()

        query = "SELECT student\_codes FROM ExamSessions WHERE session\_code = ?"

        cursor.execute(query, (session\_code,))

        result = cursor.fetchone()

        if result:

            student\_codes = result[0]

            student\_list = student\_codes.split(",") if student\_codes else []

            if not student\_list:

                return {"error": f"Ca thi {session\_code} không có sinh viên nào."}

            placeholders = ','.join('?' for \_ in student\_list)

            query\_students = f"""

                SELECT student\_code, full\_name, phone\_number, face\_photo

                FROM Students

                WHERE student\_code IN ({placeholders})

            """

            cursor.execute(query\_students, student\_list)

            student\_info\_list = cursor.fetchall()

            if student\_info\_list:

                student\_data = []

                for student\_code, full\_name, phone\_number, face\_photo in student\_info\_list:

                    student\_data.append({

                        "name": full\_name,

                        "student\_id": student\_code,

                        "phone": phone\_number if phone\_number else "Chưa cập nhật",

                        "image\_url": face\_photo if face\_photo else "/static/default\_avatar.png",

                        "status": "Đang chờ xác thực"

                    })

                return {"student\_info\_result": {"student\_info": student\_data}}

            else:

                return {"error": "Không tìm thấy thông tin sinh viên trong danh sách."}

        else:

            return {"error": f"Không tìm thấy ca thi với mã {session\_code}."}

    except sqlite3.Error as e:

        return {"error": f"Lỗi truy vấn cơ sở dữ liệu: {e}"}

    finally:

        conn.close()

Hình 3.6: Ảnh mã nguồn Mô-đun đọc danh sách từ cơ sở dữ liệu

* + 1. Giao diện chức năng so khớp thông tin sinh viên với danh sách

Giao diện hệ thống được thiết kế đơn giản và dễ sử dụng, bao gồm ba thành phần chính: khu vực tải file Excel, nơi người dùng có thể tải file và kiểm tra đường dẫn file đã chọn; khu vực kết quả trích xuất, hiển thị thông tin sinh viên như họ tên, mã số, thông tin phòng thi cùng trạng thái xử lý và các nút chức năng, cho phép thực hiện thao tác trích xuất dữ liệu sau khi tải file lên. Sự bố trí hợp lý này giúp người dùng thao tác nhanh chóng và theo dõi hiệu quả quá trình xử lý thông tin.

A screenshot of a computer

Description automatically generated

Hình 3.7: Giao diện chức năng so khớp thông tin sinh viên

* + 1. Kiểm thử chức năng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mục tiêu kiểm thử** | **Kịch bản kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** |
| 1 | Đảm bảo hệ thống có thể tải và xử lý file Excel hợp lệ. | 1. Nhấn nút "Tải file Excel". 2. Chọn file Excel hợp lệ từ máy tính. 3. Nhấn "Mở". | 1. Hệ thống hiển thị đường dẫn của file đã tải lên. 2. Đường dẫn file xuất hiện chính xác trên giao diện. | Như kết quả mong đợi |
| 2 | Đảm bảo hệ thống xử lý đúng các file không phải Excel. | 1. Nhấn nút "Tải file Excel". 2. Chọn file không phải file Excel (ví dụ: .txt, .pdf). 3. Nhấn "Mở". | Hệ thống hiển thị thông báo lỗi, yêu cầu người dùng chọn file Excel hợp lệ. | Thông báo định dạng không hợp lệ |
| 3 | Đảm bảo hệ thống trích xuất thông tin chính xác từ file Excel hợp lệ. | 1. Tải file Excel hợp lệ lên hệ thống. 2. Nhấn nút "Trích xuất thông tin". | 1. Hệ thống hiển thị các thông tin trích xuất từ file: mã sinh viên, họ tên, phòng thi, giám thị. 2. Trạng thái trích xuất thành công được hiển thị. | Các thông tin được trích xuất đúng với thực tế |
| 4 | Đảm bảo hệ thống xử lý khi không có dữ liệu trong file Excel. | 1. Tải file Excel không chứa dữ liệu hợp lệ (ví dụ, chỉ có tiêu đề cột mà không có thông tin). 2. Nhấn nút "Trích xuất thông tin". | Hệ thống hiển thị thông báo "Không nhận diện được thông tin". | Lỗi không thông báo, trả về giao diện trống |
| 5 | Đảm bảo hệ thống thông báo lỗi khi trích xuất dữ liệu sai. | 1. Tải file Excel có dữ liệu sai lệch (ví dụ: mã sinh viên sai, thiếu thông tin phòng thi). 2. Nhấn nút "Trích xuất thông tin". | Hệ thống hiển thị thông báo lỗi hoặc thông báo "Lỗi trong quá trình xử lý" nếu không thể trích xuất thông tin chính xác. | Thông báo lỗi như mong đợi |
| 6 | Đảm bảo hệ thống hiển thị thông báo trạng thái chính xác sau khi trích xuất thông tin thành công. | 1. Tải file Excel hợp lệ lên hệ thống. 2. Nhấn nút "Trích xuất thông tin". | Hệ thống hiển thị thông báo "Trích xuất thành công" trong khu vực thông báo trạng thái. | Chưa thông báo lỗi |
| 7 | Kiểm thử giao diện người dùng (UI) | Kiểm tra tính năng các nút “Tải file”, “Xử lý” trên giao diện người dùng | Giao diện dễ sử dụng, các nút chức năng hoạt động đúng và không gặp lỗi. | Giao diện dễ sử dụng, không gặp lỗi |
| 8 | Đảm bảo hệ thống hiển thị thông báo khi không có thông tin phòng thi trong file. | 1. Tải file Excel hợp lệ nhưng không có thông tin phòng thi. 2. Nhấn nút "Trích xuất thông tin". | Hệ thống hiển thị thông báo "Không có thông tin phòng thi" trong khu vực kết quả trích xuất. | Lỗi không thông báo, trả về giao diện trống |

Bảng 3.3: Kiểm thử chức năng so khớp thông tin sinh viên

* 1. Chức năng đối chiếu khuôn mặt trên thẻ sinh viên với ảnh thực tế

Chức năng này giúp xác thực danh tính sinh viên thông qua việc đối chiếu khuôn mặt trên thẻ sinh viên với hình ảnh thu được từ camera trong thời gian thực, nhằm đảm bảo tính chính xác và hạn chế gian lận.

* + 1. Xây dựng các Mô-đun liên quan
       1. Mô-đun nhận diện và cắt khuôn mặt có trên thẻ sinh viên

**Mục đích:** Mô-đun này có nhiệm vụ phát hiện và cắt khuôn mặt từ ảnh thẻ sinh viên, sau đó chuyển khuôn mặt sang dạng base64 để dễ dàng truyền qua API hoặc lưu trữ.

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả chi tiết** |
| Tham số đầu vào | |  | | --- | |  | | img\_path (str): Đường dẫn đến ảnh cần xử lý (ảnh sinh viên hoặc ảnh từ tài liệu). | | |
| Thư viện sử dụng | - cv2 (OpenCV): Dùng để xử lý ảnh, phát hiện khuôn mặt và thực hiện các phép toán hình ảnh. - logging: Dùng để ghi lại các thông báo và lỗi trong quá trình xử lý. - base64: Chuyển đổi ảnh sang chuỗi base64 để trả về cho API. - PIL: Dùng để chuyển đổi ảnh từ OpenCV sang định dạng có thể mã hóa thành base64. |
| Quy trình hoạt động | 1. Đọc ảnh từ đường dẫn img\_path sử dụng OpenCV. 2. Tiến hành phát hiện khuôn mặt trong ảnh bằng CascadeClassifier (Haar Cascade). 3. Nếu phát hiện khuôn mặt, cắt và phóng to khuôn mặt theo tỉ lệ xác định. 4. Chuyển ảnh khuôn mặt đã cắt thành định dạng base64. 5. Trả về chuỗi base64 của khuôn mặt nếu tìm thấy khuôn mặt, hoặc None nếu không tìm thấy khuôn mặt. |
| Kết quả trả về | str: Chuỗi base64 của khuôn mặt đã được cắt và mã hóa (nếu phát hiện khuôn mặt). Nếu không phát hiện khuôn mặt, trả về None. |

Bảng 3.3: Bảng mô tả Mô-đun nhận diện và cắt khuôn mặt có trên thẻ sinh viên

def process\_student\_id(*img\_path*):

    try:

        img = cv2.imread(img\_path)

        if img is None:

            raise ValueError(f"Không thể đọc được ảnh từ đường dẫn {img\_path}")

        logging.info("Đã đọc thành công ảnh.")

        face\_img = detect\_face(img)

        if face\_img is not None:

   pil\_img = Image.fromarray(cv2.cvtColor(face\_img, cv2.COLOR\_BGR2RGB))

            buffered = io.BytesIO()

            pil\_img.save(buffered, *format*="JPEG")

            img\_str = base64.b64encode(buffered.getvalue()).decode('utf-8')

            logging.info("Đã cắt và mã hóa khuôn mặt thành công.")

            return img\_str

        else:

            logging.warning("Không tìm thấy khuôn mặt.")

            return None

    except Exception as e:

        logging.error(f"Lỗi trong quá trình xử lý: {e}")

        return None

Hình 3.8: Ảnh mã nguồn Mô-đun nhận diện và cắt khuôn mặt có trên thẻ sinh viên

* + - 1. Mô-đun đối chiếu khuôn mặt đã cắt trên thẻ với khuôn mặt thực tế

**Mục đích:** Mô-đun này có nhiệm vụ phát hiện và cắt khuôn mặt từ ảnh thẻ sinh viên, sau đó chuyển khuôn mặt sang dạng base64 để dễ dàng truyền qua API hoặc lưu trữ. Mô-đun này sẽ thực hiện các bước xử lý ảnh để tìm khuôn mặt, cắt khuôn mặt ra khỏi ảnh gốc, và cuối cùng chuyển ảnh khuôn mặt sang định dạng base64 để có thể sử dụng trong các ứng dụng web hoặc các dịch vụ API.

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả chi tiết** |
| Tham số đầu vào | |  | | --- | |  | | - uploaded\_image (np.array): Ảnh tải lên dưới dạng ma trận NumPy. - face\_image\_base64 (str): Ảnh khuôn mặt đã được mã hóa dạng base64. - threshold (float): Ngưỡng so sánh (mặc định là 0.6). | | |
| Thư viện sử dụng | - base64: Dùng để giải mã chuỗi base64 thành ảnh. - numpy: Sử dụng cho các phép toán với vector và tính khoảng cách Euclidean. - cv2 (OpenCV): Dùng để xử lý ảnh, chuyển đổi ảnh giữa các định dạng. - PIL: Dùng để mở ảnh từ byte stream. - dlib: Dùng để phát hiện khuôn mặt, dự đoán các đặc trưng của khuôn mặt và trích xuất đặc trưng khuôn mặt. |
| Quy trình hoạt động | 1. Giải mã ảnh từ base64: Chuyển ảnh khuôn mặt từ chuỗi base64 về dạng ảnh. 2. Phát hiện khuôn mặt: Sử dụng dlib để phát hiện khuôn mặt trong cả ảnh tải lên và ảnh base64. 3. Trích xuất vector đặc trưng: Trích xuất vector đặc trưng của khuôn mặt bằng mô hình face\_recognition\_model\_v1 của dlib. 4. So sánh khuôn mặt: Tính khoảng cách Euclidean giữa hai vector đặc trưng, nếu khoảng cách nhỏ hơn ngưỡng threshold, kết luận là khuôn mặt giống nhau. Nếu không, kết luận là khác nhau. 5. Trả kết quả: Trả về kết quả so sánh (cùng một người hoặc khác nhau) và khoảng cách Euclidean. |
| Kết quả trả về | str: Kết quả so sánh hai khuôn mặt dưới dạng chuỗi, ví dụ: "Cùng 1 người (Khoảng cách Euclidean: 0.45)" hoặc "2 người khác nhau (Khoảng cách Euclidean: 1.20)". |

Bảng 3.4: Bảng mô tả Mô-đun đối chiếu khuôn mặt đã cắt với khuôn mặt thực tế

def compare\_faces(uploaded\_image, face\_image\_base64, threshold=0.6):

    image1 = cv2.cvtColor(uploaded\_image, cv2.COLOR\_RGB2BGR)

    image2 = decode\_base64\_image(face\_image\_base64)

    embedding1 = get\_face\_embedding(image1)

    embedding2 = get\_face\_embedding(image2)

    if embedding1 is None:

        return "Ảnh tải lên không tìm thấy khuôn mặt"

    if embedding2 is None:

        return "Ảnh khuôn mặt từ thẻ không tìm thấy khuôn mặt"

    distance = np.linalg.norm(embedding1 - embedding2)

    if distance < threshold:

        return f"Cùng 1 người (Khoảng cách Euclidean: {distance:.2f})"

    else:

        return f"2 người khác nhau (Khoảng cách Euclidean: {distance:.2f})"

Hình 3.9: Ảnh mã nguồn Mô-đun đối chiếu ảnh với khuôn mặt thực tế

* + 1. Giao diện chức năng đối chiếu khuôn mặt trên thẻ với ảnh thực tế

Giao diện người dùng sẽ là công cụ trực quan giúp cán bộ giám sát dễ dàng thực hiện quy trình đối chiếu khuôn mặt nhanh chóng, chính xác trong các kỳ thi.

A screenshot of a computer

Description automatically generated

Hình 3.10: Giao diện chức năng nhận diện khuôn mặt

* + - 1. Giới thiệu giao diện chính chức năng

Giao diện sẽ gồm các thành phần như sau:

1. Khu vực tải ảnh thẻ sinh viên và ảnh cá nhân sinh viên vào hệ thống:

* Nút tải file ảnh: Cho phép người dùng tải file ảnh lên để hệ thống xử lý.
* Hiển thị đường dẫn tải lên: Sau khi tải ảnh, hệ thống sẽ hiển thị đường dẫn file trên giao diện, giúp người dùng kiểm tra file cần xử lý.

1. Khu vực kết quả trích xuất:

* Kết quả trích xuất thông tin: Hiển thị các thông tin trích xuất từ kết quả so sánh 2 khuôn mặt của hệ thống và thông tin trên thẻ.
  + 1. Kiểm thử chức năng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mục tiêu kiểm thử** | **Kịch bản kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** |
| 1 | Đảm bảo hệ thống có thể tải và xử lý 2 file ảnh hợp lệ. | 1. Nhấn nút "Tải file". 2. Chọn file JPG, PNG, hay các file định dạng hình ảnh hợp lệ từ máy tính. 3. Nhấn "Mở". | 1. Hệ thống hiển thị đường dẫn của file đã tải lên. 2. Đường dẫn file xuất hiện chính xác trên giao diện. | Như kết quả mong đợi |
| 2 | Đảm bảo hệ thống xử lý đúng các file không thuộc định dạng hình ảnh | 1. Nhấn nút "Tải file". 2. Chọn file không phải file ảnh (ví dụ: .txt, .mp3). 3. Nhấn "Mở". | Hệ thống hiển thị thông báo lỗi, yêu cầu người dùng chọn file Excel hợp lệ. | Thông báo định dạng không hợp lệ |
| 3 | Đảm bảo hệ thống hiển thị kết quả chính xác sau khi so sánh thành công. | 1. Tải file ảnh hợp lệ lên hệ thống. 2. Nhấn nút "Gửi ảnh". | Hệ thống hiển thị ngay kết quả so sánh và thông tin thẻ sinh viên. | Chưa thông báo lỗi |
| 4 | Kiểm thử giao diện người dùng (UI) | Kiểm tra tính năng các nút “Tải file”, “Xử lý” trên giao diện người dùng | Giao diện dễ sử dụng, các nút chức năng hoạt động đúng và không gặp lỗi. | Giao diện dễ sử dụng, không gặp lỗi |
| 5 | Đảm bảo hệ thống hiển thị thông báo khi chương trình gặp lỗi | 1. Nhấn nút “Tải file”. 2. Nhấn nút "Gửi ảnh". | Hệ thống hiển thị thông báo "Không xử lý được" trong khu vực kết quả trích xuất. | Lỗi không thông báo, trả về giao diện trống |

Bảng 3.4: Kiểm thử chức năng đối chiếu khuôn mặt với ảnh thực tế

* 1. Chức năng xuất phiếu và giám sát quy trình thi

Chức năng xuất phiếu và giám sát quy trình thi đóng vai trò quan trọng trong việc quản lý kỳ thi của sinh viên, đảm bảo tính minh bạch và công bằng trong suốt quá trình thi. Hệ thống sẽ tự động tạo phiếu thi cho từng sinh viên, theo dõi việc tham gia thi và giám sát hành vi của sinh viên trong phòng thi để phát hiện kịp thời các hành vi gian lận. Các mô-đun sẽ được triển khai để kết nối và xử lý thông tin liên quan đến sinh viên, kỳ thi, giám thị, và các hành vi trong phòng thi.

* + 1. Xây dựng các Mô-đun liên quan
       1. Mô-đun in phiếu thi

Mô-đun in phiếu thi có nhiệm vụ tạo ra các phiếu dự thi cho sinh viên dưới dạng file PDF. Chức năng này hỗ trợ việc cấp phát phiếu thi, bao gồm các thông tin cần thiết như tên sinh viên, mã sinh viên, môn thi, mã khóa và vị trí ngồi. Hệ thống sử dụng thư viện FPDF để tạo file PDF, kết hợp với font chữ tùy chỉnh để đảm bảo giao diện đẹp mắt và dễ đọc. Phiếu thi được lưu trữ vào thư mục định sẵn và trả về kết quả với thông tin file PDF đã tạo. Quy trình hoàn thiện mô-đun này giúp đơn giản hóa công tác cấp phát phiếu thi và bảo mật thông tin thi cử.

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả chi tiết** |
| Tham số đầu vào | |  | | --- | |  | | student\_name (str): Tên sinh viên, student\_msv (str): Mã sinh viên, exam\_name (str): Tên môn thi, exam\_code (str): Mã khóa, seat\_position (str): Vị trí ngồi, output\_dir (str): Thư mục xuất file (mặc định là "tickets") | | |
| Thư viện sử dụng | FPDF: Tạo file PDF. |
| Quy trình hoạt động | 1. Kiểm tra thư mục font và tạo thư mục xuất file nếu cần.  2. Tạo file PDF, thêm các thông tin cần thiết vào phiếu thi.  3. Lưu file PDF vào thư mục đã định và trả về thông tin kết quả. |
| Kết quả trả về | Một từ điển chứa status (thành công hay lỗi), message (thông báo), ticket\_file (đường dẫn file PDF) và ticket\_info (danh sách thông tin phiếu thi). |

Bảng 3.5: Bảng mô tả Mô-đun in phiếu thi cho sinh viên

#module.generate\_ticket.py

import random

import string

from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

import os

def generate\_random\_string(*length*=8):

    characters = string.ascii\_letters + string.digits

    return ''.join(random.choice(characters) for i in range(length))

def generate\_exam\_ticket(*student\_name*, *student\_id*, *output\_path*):

    seating\_position = random.randint(1, 20)

    exam\_account = generate\_random\_string()

    exam\_password = generate\_random\_string()

    ticket\_filename = f"{student\_id}\_exam\_ticket.pdf"

    ticket\_path = os.path.join(output\_path, ticket\_filename)

    c = canvas.Canvas(ticket\_path, *pagesize*=letter)

    width, height = letter

    c.setFont("Helvetica-Bold", 18)

    c.drawString(200, height - 40, "Phiếu Thi Sinh Viên")

    c.setFont("Helvetica", 12)

    c.drawString(50, height - 80, f"Tên Sinh Viên: {student\_name}")

    c.drawString(50, height - 100, f"Mã Sinh Viên: {student\_id}")

    c.drawString(50, height - 120, f"Vị trí ngồi: {seating\_position}")

    c.drawString(50, height - 140, f"Tài khoản thi: {exam\_account}")

    c.drawString(50, height - 160, f"Mật khẩu thi: {exam\_password}")

    c.save()

    return ticket\_path

Hình 3.11: Ảnh mã nguồn Mô-đun in phiếu thi cho sinh viên

* 1. Các chức năng quản lý của quản trị viên

Chức năng quản lý hỗ trợ quản trị viên thực hiện các thao tác quản lý toàn diện và chính xác. Quản trị viên có thể quản lý thông tin sinh viên thông qua các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm theo mã sinh viên, tên, lớp học hoặc chuyên ngành, đồng thời theo dõi trạng thái tham gia kỳ thi, đồng bộ phiếu thi, lịch thi và quản lý kết quả thi cùng các hành vi vi phạm. Bên cạnh đó, chức năng quản lý giám thị hỗ trợ quản trị viên quản lý thông tin giám thị, phân công giám thị vào phòng thi, theo dõi lịch trình làm việc và cập nhật kết quả giám sát để đảm bảo tính minh bạch. Với chức năng quản lý phòng thi, quản trị viên có thể thực hiện các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm thông tin phòng thi dựa trên mã phòng, địa điểm, sức chứa, cùng việc phân bổ thí sinh và giám thị một cách khoa học. Sự tích hợp chặt chẽ giữa các chức năng này giúp tối ưu hóa quy trình tổ chức, quản lý và giám sát kỳ thi, đảm bảo công bằng và hiệu quả trong toàn bộ hệ thống.

A screenshot of a computer

Description automatically generated

Hình 3.12: Giao diện trang chủ trang quản trị viên

* + 1. Chức năng quản lý sinh viên

Chức năng quản lý sinh viên trong hệ thống quản lý kỳ thi hỗ trợ quản trị viên theo dõi và quản lý thông tin chi tiết của sinh viên tham gia kỳ thi, bao gồm các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm sinh viên theo các tiêu chí như mã sinh viên, tên và lớp học. Hệ thống cung cấp khả năng theo dõi trạng thái tham gia thi của sinh viên, đồng bộ hóa thông tin với các phiếu thi và lịch thi, cũng như lưu trữ kết quả thi. Chức năng này đảm bảo việc quản lý sinh viên một cách khoa học, chính xác và minh bạch, góp phần hỗ trợ tổ chức và giám sát kỳ thi hiệu quả.

* + - 1. Giao diện chức năng quản lý sinh viên

A screenshot of a computer

Description automatically generated

Hình 3.13: Giao diện trang quản lý sinh viên

* + 1. Chức năng quản lý giám thị

Chức năng quản lý giám thị trong hệ thống quản lý kỳ thi giúp quản trị viên theo dõi và tổ chức thông tin liên quan đến giám thị một cách hiệu quả. Quản trị viên có thể thực hiện các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm giám thị dựa trên các tiêu chí như mã giám thị, tên hoặc phòng thi được phân công. Hệ thống hỗ trợ việc phân công giám thị vào các phòng thi, quản lý lịch trình làm việc của giám thị và cập nhật thông tin về kết quả giám sát trong suốt kỳ thi. Chức năng này không chỉ đảm bảo tính minh bạch và công bằng trong quá trình giám sát kỳ thi mà còn tối ưu hóa việc phân bổ nhân sự, hỗ trợ quản lý hiệu quả và giảm thiểu sai sót trong việc tổ chức kỳ thi.

* + - 1. Các thành phần liên quan:

Proctor Entity:

@Entity  
@Table(name = "Proctors")  
public class Proctor {  
  
 @Id  
 @Column(name = "proctor\_code", nullable = false, unique = true)  
 private String proctorCode;  
  
 @Column(nullable = false)  
 private String fullName;  
  
 @Column(nullable = false)  
 private String department;  
  
 @Column(nullable = true)  
 private String phoneNumber;  
  
 @Column(nullable = true)  
 private String email;  
  
 public Proctor() {}  
  
 public Proctor(String proctorCode, String fullName, String department, String phoneNumber, String email) {  
 this.proctorCode = proctorCode;  
 this.fullName = fullName;  
 this.department = department;  
 this.phoneNumber = phoneNumber;  
 this.email = email;  
 }  
  
 // Getters và Setters  
 public String getProctorCode() {  
 return proctorCode;  
 }  
  
 public void setProctorCode(String proctorCode) {  
 this.proctorCode = proctorCode;  
 }  
  
 public String getFullName() {  
 return fullName;  
 }  
  
 public void setFullName(String fullName) {  
 this.fullName = fullName;  
 }  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 @Override  
 public String toString() {  
 return "Proctor{" +  
 "proctorCode='" + proctorCode + '\'' +  
 ", fullName='" + fullName + '\'' +  
 ", department='" + department + '\'' +  
 ", phoneNumber='" + phoneNumber + '\'' +  
 ", email='" + email + '\'' +  
 '}';  
 }  
}

Hình 3.14: Ảnh mã nguồn Proctor Entity

Proctor Repository:

package com.personal\_project.exam\_management\_system.repository;  
  
import com.personal\_project.exam\_management\_system.entity.Proctor;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.Optional;  
  
public interface ProctorRepository extends JpaRepository<Proctor, Long> {  
 Optional<Proctor> findByProctorCode(String proctorCode);  
}

Hình 3.15: Ảnh mã nguồn Proctor Repository

Proctor Service:

@Service  
public class ProctorService {  
  
 @Autowired  
 private ProctorRepository proctorRepository;  
  
 public Proctor findByProctorCode(String proctorCode) {  
 return proctorRepository.findByProctorCode(proctorCode)  
 .orElseThrow(() -> new RuntimeException("Proctor not found with code: " + proctorCode));  
 }  
  
 public Proctor addProctor(Proctor proctor) {  
 return proctorRepository.save(proctor);  
 }  
  
 public Proctor updateProctor(Proctor proctor) {  
 return proctorRepository.save(proctor);  
 }  
  
 public List<Proctor> getAllProctors() {  
 return proctorRepository.findAll();  
 }  
}

Hình 3.16: Ảnh mã nguồn Proctor Service

Proctor Controller:

@Controller  
@RequestMapping("/proctor")  
public class ProctorController {  
 // Trang dashboard-proctor  
 @GetMapping("/dashboard-proctor")  
 public String dashboardProctor(Model model) {  
 String userCode = userService.getUserCode();  
  
 if (userCode != null) {  
 Proctor proctor = proctorService.findByProctorCode(userCode);  
  
 if (proctor != null) {  
 model.addAttribute("proctor", proctor);  
 model.addAttribute("pageContent", "dashboard-proctor");  
 return "template-proctor";  
 } else {  
 model.addAttribute("error", "Proctor not found");  
 return "redirect:error";  
 }  
 } else {  
 model.addAttribute("error", "User code not found");  
 return "redirect:error";  
 }  
 }  
  
 // Trang thông tin cá nhân của giám thị  
 @GetMapping("/info")  
 public String proctorInfo(Model model) {  
 String userCode = userService.getUserCode();  
  
 if (userCode != null) {  
 Proctor proctor = proctorService.findByProctorCode(userCode);  
  
 if (proctor != null) {  
 model.addAttribute("proctor", proctor);  
 model.addAttribute("pageContent", "proctor-info");  
 return "template-proctor";  
 } else {  
 model.addAttribute("error", "Proctor not found");  
 return "redirect:error";  
 }  
 } else {  
 model.addAttribute("error", "User code not found");  
 return "redirect:error";  
 }  
 }  
  
 @GetMapping("/schedule")  
 public String viewProctorExamSchedule(Model model) {  
 String proctorCode = userService.getUserCode();  
 Optional<Proctor> proctor = proctorRepository.findByProctorCode(proctorCode);  
  
 if (proctor.isEmpty()) {  
 model.addAttribute("error", "Giám thị không tồn tại");  
 return "redirect:error";  
 }  
 List<ExamSession> examSessions = examSessionRepository.findByProctorCode(proctorCode);  
  
 model.addAttribute("proctor", proctor.get());  
 model.addAttribute("examSessions", examSessions);  
  
 model.addAttribute("pageContent", "exam-schedule-proctor");  
 return "template-proctor";  
 }  
  
 @GetMapping("/start-exam-session")  
 public String startExamSession(Model model) {  
 return "start-exam-session";  
 }  
  
 @PostMapping("/start-exam-session")  
 @ResponseBody // Đảm bảo trả về dữ liệu JSON thay vì view  
 public ResponseEntity<?> startExamSession(@RequestBody Map<String, String> payload) {  
 String sessionCode = payload.get("sessionCode");  
 String url = "http://127.0.0.1:8000/get-student-info/";  
  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.*APPLICATION\_JSON*);  
  
 HttpEntity<String> entity = new HttpEntity<>("{\"sessionCode\":\"" + sessionCode + "\"}", headers);  
  
 try {  
 ResponseEntity<String> response = appConfig.restTemplate().exchange(url, HttpMethod.*POST*, entity, String.class);  
  
 if (response.getStatusCode() == HttpStatus.*OK*) {  
 return ResponseEntity.*ok*(response.getBody()); // Trả về JSON từ Flask  
 } else {  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("{\"error\": \"Không lấy được danh sách sinh viên\"}");  
 }  
 } catch (Exception e) {  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("{\"error\": \"Lỗi kết nối đến Flask API\"}");  
 }  
 }  
  
 @GetMapping("/verify-face")  
 public String verifyFace(@RequestParam("student\_id") String studentId,  
 @RequestParam("name") String name,  
 @RequestParam("phone") String phone,  
 @RequestParam("image\_url") String imageUrl,  
 Model model) {  
 model.addAttribute("studentId", studentId);  
 model.addAttribute("name", name);  
 model.addAttribute("phone", phone);  
 model.addAttribute("imageUrl", imageUrl);  
 return "verify-face";  
 }  
  
 @GetMapping("/virtual-exam")  
 public String virtualExam(Model model) {  
 return "virtual-exam";  
 }  
}

Hình 3.17: Ảnh mã nguồn Proctor Controller

* + - 1. Giao diện của trang quản lý giám thị

A screenshot of a computer

Description automatically generated

Hình 3.18: Giao diện trang quản lý giám thị

* + 1. Chức năng quản lý phòng thi

Chức năng quản lý phòng thi trong hệ thống quản lý kỳ thi hỗ trợ quản trị viên tổ chức và giám sát thông tin phòng thi một cách hiệu quả. Quản trị viên có thể thực hiện các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm phòng thi dựa trên các tiêu chí như mã phòng, địa điểm, và sức chứa. Hệ thống cho phép phân bổ thí sinh và giám thị vào từng phòng thi, theo dõi trạng thái sử dụng phòng thi, và quản lý lịch thi trong mỗi phòng. Ngoài ra, chức năng này còn hỗ trợ cập nhật kết quả thi và lưu trữ thông tin liên quan để đảm bảo tổ chức kỳ thi diễn ra minh bạch, khoa học và tối ưu hóa việc sử dụng nguồn lực phòng thi.

* + - 1. Các thành phần liên quan

ExamRoom Entity:

@Entity  
@Table(name = "examrooms")  
public class ExamRoom {  
  
 @Id  
 @Column(name = "room\_code", nullable = false)  
 private String roomCode;  
  
  
 @Column(name = "address", nullable = false)  
 private String address;  
  
 @Column(name = "max\_capacity" ,nullable = false)  
 private int maxCapacity;  
  
 public ExamRoom() {}  
  
 public ExamRoom(String roomCode, String address, int maxCapacity) {  
 this.roomCode = roomCode;  
 this.address = address;  
 this.maxCapacity = maxCapacity;  
 }  
  
 // Getters and Setters  
 public String getRoomCode() {  
 return roomCode;  
 }  
  
 public void setRoomCode(String roomCode) {  
 this.roomCode = roomCode;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public int getMaxCapacity() {  
 return maxCapacity;  
 }  
  
 public void setMaxCapacity(int maxCapacity) {  
 this.maxCapacity = maxCapacity;  
 }  
}

Hình 3.19: Ảnh mã nguồn ExamRoom Entity

ExamRoom Repository:

package com.personal\_project.exam\_management\_system.repository;  
  
import com.personal\_project.exam\_management\_system.entity.ExamRoom;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ExamRoomRepository extends JpaRepository<ExamRoom, String> {  
}

Hình 3.20: Ảnh mã nguồn ExamRoom Repository

ExamRoom Service:

@Service  
public class ExamRoomService {  
  
 @Autowired  
 private ExamRoomRepository examRoomRepository;  
  
 public List<ExamRoom> getAllRooms() {  
 return examRoomRepository.findAll();  
 }  
  
 public Optional<ExamRoom> getRoomByCode(String roomCode) {  
 return examRoomRepository.findById(roomCode);  
 }  
  
 public void addRoom(ExamRoom examRoom) {  
 examRoomRepository.save(examRoom);  
 }  
  
 public void deleteRoom(String roomCode) {  
 examRoomRepository.deleteById(roomCode);  
 }  
  
 public void updateRoom(String roomCode, ExamRoom updatedRoom) {  
 examRoomRepository.findById(roomCode).map(room -> {  
 room.setAddress(updatedRoom.getAddress());  
 room.setMaxCapacity(updatedRoom.getMaxCapacity());  
 return examRoomRepository.save(room);  
 }).orElseThrow(() -> new RuntimeException("Room not found"));  
 }  
}

Hình 3.21: Ảnh mã nguồn ExamRoom Service

ExamRoom Controller:

@GetMapping("/manage-examrooms")  
public String getAllRooms(Model model) {  
 List<ExamRoom> rooms = examRoomService.getAllRooms();  
 model.addAttribute("rooms", rooms);  
 model.addAttribute("pageContent", "manage-examrooms");  
 return "template-admin";  
}  
  
@GetMapping("/exam-rooms/{roomCode}")  
public String getRoomByCode(@PathVariable String roomCode, Model model) {  
 Optional<ExamRoom> room = examRoomService.getRoomByCode(roomCode);  
 if (room.isPresent()) {  
 model.addAttribute("room", room.get());  
 return "edit-examroom";  
 } else {  
 return "redirect:/admin/manage-examrooms";  
 }  
}  
  
@GetMapping("/exam-rooms/add")  
public String showAddExamRoomForm(Model model) {  
 model.addAttribute("examRoom", new ExamRoom());  
 return "create-examroom";  
}  
  
@PostMapping("/exam-rooms/add")  
public String addExamRoom(@ModelAttribute("examRoom") ExamRoom examRoom) {  
 examRoomService.addRoom(examRoom);  
 return "redirect:/admin/manage-examrooms";  
}  
  
@PutMapping("/exam-rooms/{roomCode}")  
public String updateRoom(@PathVariable String roomCode, @ModelAttribute ExamRoom updatedRoom) {  
 examRoomService.updateRoom(roomCode, updatedRoom);  
 return "redirect:/admin/exam-rooms";  
}  
  
@DeleteMapping("/exam-rooms/{roomCode}")  
public String deleteRoom(@PathVariable String roomCode) {  
 examRoomService.deleteRoom(roomCode);  
 return "redirect:/admin/exam-rooms";  
}

Hình 3.22: Ảnh mã nguồn ExamRoom Controller

* + - 1. Giao diện chức năng quản lý phòng thi

A screenshot of a computer

Description automatically generated

Hình 3.23: Giao diện trang quản lý phòng thi

* + 1. Chức năng quản lý học phần

Chức năng quản lý học phần trong hệ thống quản lý kỳ thi hỗ trợ quản trị viên tổ chức và giám sát thông tin phòng thi một cách hiệu quả. Quản trị viên có thể thực hiện các thao tác như thêm mới, chỉnh sửa, xóa và tìm kiếm phòng thi dựa trên các tiêu chí như mã phòng, địa điểm, và sức chứa. Hệ thống cho phép phân bổ thí sinh và giám thị vào từng phòng thi, theo dõi trạng thái sử dụng phòng thi, và quản lý lịch thi trong mỗi phòng. Ngoài ra, chức năng này còn hỗ trợ cập nhật kết quả thi và lưu trữ thông tin liên quan để đảm bảo tổ chức kỳ thi diễn ra minh bạch, khoa học và tối ưu hóa việc sử dụng nguồn lực phòng thi.

* + - 1. Các thành phần liên quan

Course Entity:

@Entity  
@Table(name = "Courses")  
public class Course {  
  
 @Id  
 @Column(name = "course\_code")  
 private String courseCode;  
  
 @Column(nullable = false)  
 private String courseName;  
  
 @Column(nullable = false)  
 private int creditHours;  
  
 @Column(nullable = true)  
 private String description;  
  
 @Column(nullable = false)  
 private String department;  
  
 @Column(nullable = false)  
 private String semester;  
  
 @Column(nullable = false)  
 private int year;  
  
 public Course() {}  
  
 public Course(String courseCode, String courseName, int creditHours, String description, String department, String semester, int year) {  
 this.courseCode = courseCode;  
 this.courseName = courseName;  
 this.creditHours = creditHours;  
 this.description = description;  
 this.department = department;  
 this.semester = semester;  
 this.year = year;  
 }

Hình 3.24: Ảnh mã nguồn Course Entity

Course Repository:

@Repository  
public interface CourseRepository extends JpaRepository<Course, Long> {  
}

Hình 3.25: Ảnh mã nguồn Course Repository

Course Service:

@Service  
public class CourseService {  
  
 private final CourseRepository courseRepository;  
  
 @Autowired  
 public CourseService(CourseRepository courseRepository) {  
 this.courseRepository = courseRepository;  
 }  
  
 public List<Course> getAllCourses() {  
 return courseRepository.findAll();  
 }  
  
 public Optional<Course> getCourseByCode(String courseCode) {  
 return courseRepository.findAll().stream()  
 .filter(course -> course.getCourseCode().equals(courseCode))  
 .findFirst();  
 }  
  
 public void addCourse(Course course) {  
 courseRepository.save(course);  
 }  
  
 public void updateCourse(Course course) {  
 courseRepository.save(course);  
 }  
  
 public void deleteCourse(Long courseId) {  
 courseRepository.deleteById(courseId);  
 }  
}

Hình 3.26: Ảnh mã nguồn Course Service

Course Controller:

@GetMapping("/manage-courses")  
public String getAllCourses(Model model) {  
 model.addAttribute("courses", courseService.getAllCourses());  
 model.addAttribute("pageContent", "manage-courses");  
 return "template-admin";  
}  
  
@GetMapping("/manage-courses/course/create")  
public String showAddCourseForm(Model model) {  
 model.addAttribute("course", new Course());  
 return "create-course";  
}  
  
@PostMapping("/manage-courses/course/create")  
public String addCourse(@ModelAttribute("course") Course course) {  
 courseService.addCourse(course);  
 return "redirect:/admin/manage-courses";  
}  
  
@GetMapping("/courses/edit/{courseCode}")  
public String showEditCourseForm(@PathVariable("courseCode") String courseCode, Model model) {  
 courseService.getAllCourses().stream()  
 .filter(c -> c.getCourseCode().equals(courseCode))  
 .findFirst().ifPresent(course -> model.addAttribute("course", course));  
 return "edit-course";  
}  
  
@PostMapping("/courses/edit")  
public String updateCourse(@ModelAttribute("course") Course course) {  
 courseService.updateCourse(course);  
 return "redirect:/admin/manage-courses";  
}  
  
@GetMapping("/courses/delete/{courseCode}")  
public String deleteCourse(@PathVariable("courseCode") String courseCode) {  
 courseService.deleteCourse(Long.*valueOf*(courseCode));  
 return "redirect:/admin/manage-courses";  
}

Hình 3.27: Ảnh mã nguồn Course Service

* + - 1. Giao diện chức năng quản lý học phần

A screenshot of a computer

Description automatically generated

Hình 3.28: Giao diện trang quản lý học phần

* + 1. Chức năng quản lý ca thi
       1. Các thành phần liên quan

ExamSession Entity:

@Entity  
@Table(name = "examsessions")  
public class ExamSession {  
  
 @Id  
 @Column(name = "session\_code")  
 private String sessionCode;  
  
 @Column(name = "course\_code", nullable = false)  
 private String courseCode;  
  
 @Column(name = "room\_code", nullable = false)  
 private String roomCode;  
  
 @Column(name = "student\_codes")  
 private String studentCodes;  
  
 @Column(name = "proctor\_code", nullable = false)  
 private String proctorCode;  
  
 @Column(name = "exam\_date\_time", nullable = false)  
 private String examDateTime;  
  
 @Column(name = "session\_status", nullable = false)  
 private String sessionStatus;  
  
 @Column(name = "report\_file\_path")  
 private String reportFilePath;  
  
 public ExamSession() {}  
  
 public ExamSession(String sessionCode, String courseCode, String roomCode, String studentCodes,  
 String proctorCode, String examDateTime, String sessionStatus, String reportFilePath) {  
 this.sessionCode = sessionCode;  
 this.courseCode = courseCode;  
 this.roomCode = roomCode;  
 this.studentCodes = studentCodes;  
 this.proctorCode = proctorCode;  
 this.examDateTime = examDateTime;  
 this.sessionStatus = sessionStatus;  
 this.reportFilePath = reportFilePath;  
 }

Hình 3.29: Ảnh mã nguồn ExamSession Entity

ExamSession Repository:

public interface ExamSessionRepository extends JpaRepository<ExamSession, String> {  
 @Query("SELECT es FROM ExamSession es WHERE es.studentCodes LIKE %:studentCode%")  
 List<ExamSession> findByStudentCode(@Param("studentCode") String studentCode);  
  
 @Query("SELECT es FROM ExamSession es WHERE es.proctorCode LIKE %:proctorCode%")  
 List<ExamSession> findByProctorCode(@Param("proctorCode") String proctorCode);  
  
 ExamSession findBySessionCode(String sessionCode);  
  
 @Query(value = "SELECT " +  
 "es.session\_code, " +  
 "c.course\_name, " +  
 "er.address, " +  
 "GROUP\_CONCAT(s.full\_name) AS list\_student\_names, " +  
 "p.full\_name AS proctor\_name, " +  
 "es.exam\_date\_time, " +  
 "es.session\_status, " +  
 "es.report\_file\_path " +  
 "FROM ExamSessions es " +  
 "JOIN Courses c ON es.course\_code = c.course\_code " +  
 "JOIN ExamRooms er ON es.room\_code = er.room\_code " +  
 "JOIN Proctors p ON es.proctor\_code = p.proctor\_code " +  
 "JOIN Students s ON (',' || es.student\_codes || ',') LIKE ('%,' || s.student\_code || ',%') " +  
 "WHERE es.session\_code = :sessionCode " +  
 "GROUP BY es.session\_code, c.course\_name, er.address, p.full\_name, es.exam\_date\_time, es.session\_status, es.report\_file\_path",  
 nativeQuery = true)  
 List<Object[]> getExamSessionDetailsRaw(@Param("sessionCode") String sessionCode);  
}

Hình 3.30: Ảnh mã nguồn ExamSession Repository

ExamSession Service:

@Service  
public class ExamSessionService {  
  
 @Autowired  
 private ExamSessionRepository examSessionRepository;  
  
 public List<ExamSession> getAllExamSessions() {  
 return examSessionRepository.findAll();  
 }  
  
 public Optional<ExamSession> getExamSessionById(String sessionCode) {  
 return examSessionRepository.findById(sessionCode);  
 }  
  
 public ExamSessionDetailDTO getExamSessionDetails(String sessionCode) {  
 List<Object[]> results = examSessionRepository.getExamSessionDetailsRaw(sessionCode);  
  
 if (!results.isEmpty()) {  
 Object[] firstRow = results.getFirst();  
  
 String sessionCodeFromDb = (String) firstRow[0];  
 String courseName = (String) firstRow[1];  
 String address = (String) firstRow[2];  
 String listStudentNames = (String) firstRow[3]; // Đây là chuỗi chứa danh sách tên sinh viên  
 String proctorName = (String) firstRow[4];  
 String examDateTime = (String) firstRow[5];  
 String sessionStatus = (String) firstRow[6];  
 String reportFilePath = (String) firstRow[7];  
  
 return new ExamSessionDetailDTO(sessionCodeFromDb, courseName, address, listStudentNames, proctorName, examDateTime, sessionStatus, reportFilePath);  
 }  
  
 return null;  
 }  
  
 public void save(ExamSession examSession) {  
 examSessionRepository.save(examSession);  
 }  
  
 public ExamSession getSessionByCode(String sessionCode) {  
 return examSessionRepository.findBySessionCode(sessionCode);  
 }  
}

Hình 3.31: Ảnh mã nguồn ExamSession Service

ExamSession Controller:

@GetMapping("/manage-exam\_sessions")  
public String manageExamSessions(Model model) {  
 model.addAttribute("examSessions", examSessionService.getAllExamSessions());  
 model.addAttribute("pageContent", "manage-exam\_sessions");  
 return "template-admin";  
}  
  
@GetMapping("/manage-exam\_sessions/exam\_session/{sessionCode}")  
public String getExamSessionDetail(@PathVariable String sessionCode, Model model) {  
 ExamSessionDetailDTO examSessionDetails = examSessionService.getExamSessionDetails(sessionCode);  
  
 if (examSessionDetails != null) {  
 System.*out*.println("Exam Session Details: " + examSessionDetails);  
 model.addAttribute("examSessionDetails", examSessionDetails); // Đúng với giao diện HTML  
 } else {  
 model.addAttribute("errorMessage", "Không tìm thấy thông tin ca thi.");  
 }  
 model.addAttribute("pageContent", "exam\_session-detail");  
 return "template-admin";  
}  
  
@GetMapping("/manage-exam\_sessions/create")  
public String showCreateExamSessionPage(Model model) {  
 List<ExamRoom> examRooms = examRoomService.getAllRooms();  
 List<Course> courses = courseService.getAllCourses();  
 List<Proctor> proctors = proctorService.getAllProctors();  
  
 model.addAttribute("examRooms", examRooms);  
 model.addAttribute("courses", courses);  
 model.addAttribute("proctors", proctors);  
  
 return "create-exam\_session";  
}  
  
@PostMapping("/manage-exam\_sessions/create")  
public String createExamSessions(@RequestParam List<String> selectedRoomCodes,  
 @RequestParam List<String> selectedProctorCodes,  
 @RequestParam String courseCode,  
 @RequestParam String examDateTime) {  
  
 if (selectedRoomCodes.size() != selectedProctorCodes.size()) {  
 return "redirect:/error"; // Redirect về trang lỗi nếu không đúng  
 }  
  
 List<Student> allStudents = studentService.findAllStudents();  
 List<String> studentCodes = allStudents.stream()  
 .map(Student::getStudentCode)  
 .collect(Collectors.*toList*());  
  
 Random rand = new Random();  
 int totalRooms = selectedRoomCodes.size();  
 int studentsPerRoom = studentCodes.size() / totalRooms;  
  
 for (int i = 0; i < totalRooms; i++) {  
 ExamSession examSession = new ExamSession();  
 examSession.setSessionCode("SESSION-" + selectedRoomCodes.get(i) +"-"+ selectedProctorCodes.get(i));  
 examSession.setCourseCode(courseCode);  
 examSession.setRoomCode(String.*valueOf*(Collections.*singletonList*(selectedRoomCodes.get(i))));  
 examSession.setProctorCode(selectedProctorCodes.get(i));  
 examSession.setExamDateTime(examDateTime);  
 examSession.setSessionStatus("Scheduled");  
 examSession.setReportFilePath(null);  
  
 List<String> randomStudentCodes = getRandomStudents(studentCodes, studentsPerRoom);  
 examSession.setStudentCodes(String.*join*(",", randomStudentCodes));  
  
 examSessionService.save(examSession);  
  
 studentCodes.removeAll(randomStudentCodes);  
 }  
 return "redirect:/admin/manage-exam\_sessions";  
}  
  
private List<String> getRandomStudents(List<String> students, int count) {  
 List<String> randomStudents = new ArrayList<>();  
 Random rand = new Random();  
 for (int i = 0; i < count; i++) {  
 if (students.isEmpty()) break;  
 int index = rand.nextInt(students.size());  
 randomStudents.add(students.get(index));  
 students.remove(index);  
 }  
 return randomStudents;  
}

Hình 3.32: Ảnh mã nguồn ExamSession Controller

* + - 1. Giao diện chức năng quản lý ca thi

A screenshot of a computer

Description automatically generated

Hình 3.33: Giao diện trang quản lý phòng thi

KẾT LUẬN

Kết quả đạt được và hướng phát triển

Sau hơn 4 tháng triển khai, chúng tôi đã đạt được một số kết quả đáng ghi nhận, phần nào hoàn thành các mục tiêu ban đầu của dự án. Hệ thống đã được xây dựng và vận hành ổn định với các chức năng cơ bản đáp ứng nhu cầu chính như quản lý thông tin, xử lý dữ liệu và hỗ trợ quy trình xác minh trong triển khai thi cử. Các tính năng chính đã được triển khai hiệu quả, giúp tiết kiệm thời gian, giảm thiểu sai sót và nâng cao độ chính xác. Tuy nhiên, do kinh nghiệm và thời gian còn hạn chế, đề tài vẫn chưa hoàn thiện toàn diện. Thông qua đề tài chúng tôi đã được củng cố kiến thức của học phần Công nghệ Phần mềm, nắm được cách xây dựng một phầm mềm trong thực tế.

Chúng tôi xác định đề tài này rất có tiềm năng ứng dụng trong thực tế nên trong thời gian tới dự định sẽ là tiếp tục hoàn thiện và mở rộng hệ thống. Các hướng cải tiến bao gồm tối ưu hóa hiệu năng khi xử lý lượng dữ liệu lớn, bổ sung các tính năng nâng cao như báo cáo phân tích chi tiết, và cải thiện giao diện để tăng tính linh hoạt, thân thiện với người dùng. Đồng thời, việc xây dựng quy trình bảo trì và nâng cấp dài hạn cũng là mục tiêu để đảm bảo hệ thống hoạt động ổn định trong tương lai.

Những tồn tại và hạn chế

Mặc dù đã đạt được nhiều kết quả tích cực, dự án vẫn còn tồn tại một số hạn chế đáng lưu ý. Hệ thống hiện tại mới chỉ đáp ứng được các yêu cầu cơ bản, thiếu các chức năng mở rộng phù hợp với nhu cầu đặc thù. Hiệu năng có dấu hiệu suy giảm khi lượng dữ liệu tăng lên, và khả năng mở rộng chưa được tối ưu. Ngoài ra, giao diện người dùng vẫn còn đơn giản, chưa hỗ trợ tùy chỉnh theo yêu cầu cụ thể.

Những hạn chế này là bài học kinh nghiệm quý báu, đồng thời cũng là động lực để nhóm nghiên cứu tiếp tục cải tiến, hoàn thiện dự án trong thời gian tới.

TÀI LIỆU THAM KHẢO

[1] Lê Văn Phùng, *Kỹ nghệ phần mềm*, NXB Thông tin và Truyền thông, 2014.

[2] Lê Văn Phùng, Lê Hương Giang, *Kỹ nghệ phần mềm nâng cao*, NXB Thông tin và truyền thông, 2015.

[3] Dương Hữu Thành, Trường Đại học Mở Tp HCM, *Công nghệ phần mềm*, NXB Thông tin và truyền thông, 2024.

[4] OpenCV, [[*OpenCV - Open Computer Vision Library*](https://opencv.org/),](https://opencv.org/)2024.

[5] PaddleOCR, [*https://github.com/PaddlePaddle/PaddleOCR*](https://github.com/PaddlePaddle/PaddleOCR), 2024

[6] Java Spring Boot, [*https://spring.io/projects/spring-boot*](https://spring.io/projects/spring-boot), 2024

[7] Murtaza's Workshop - Robotics and AI*,* [*Face Recognition with Real Time Database*](https://www.youtube.com/watch?v=iBomaK2ARyI)*, Computer Vision,* 2022.

[8] Javapocalypse*,* [*Real time face detection and face recognition*](https://www.youtube.com/playlist?list=PLyDH8KT4GrNd7KMFIgHuxpcklUJyfk_ov)*,* 2020.

[9] Scott Tilley, Harry J. Rosenblatt, *Systems Analys and Design*, Shelly Cashman Series, 11th Edition, 2016.