ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Quý Dương

XÂY DỰNG HỆ THỐNG ĐIỂM DANH SỬ DỤNG CÔNG NGHỆ FACENET

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Khoa học máy tính

HÀ NỘI -2025

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Quý Dương

XÂY DỰNG HỆ THỐNG ĐIỂM DANH SỬ DỤNG CÔNG NGHỆ FACENET

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành:Khoa học máy tính

Cán bộ hướng dẫn: TS. Ma Thị Châu

HÀ NỘI -202

TÓM TẮT

Tóm tắt: Việc điểm danh thủ công trong các môi trường giáo dục và tổ chức thường tốn thời gian, dễ xảy ra sai sót và gian lận. Để giải quyết những hạn chế này, khóa luận này trình bày việc phân tích, thiết kế và xây dựng một "Ứng dụng điểm danh bằng nhận diện khuôn mặt sử dụng Facenet". Hệ thống được phát triển nhằm mục đích tự động hóa quy trình điểm danh, nâng cao độ chính xác và hiệu quả. Kiến trúc hệ thống bao gồm một ứng dụng di động phía client được xây dựng bằng Flutter và một dịch vụ backend phía server sử dụng Python với framework Flask. Backend chịu trách nhiệm xử lý logic nghiệp vụ, quản lý cơ sở dữ liệu thông qua SQLAlchemy và thực hiện nhận diện khuôn mặt. Người dùng (sinh viên) có thể đăng ký tài khoản, tải lên hình ảnh khuôn mặt ban đầu và tham gia vào các lớp học thông qua ứng dụng di động. Chức năng điểm danh cốt lõi cho phép người dùng chụp ảnh khuôn mặt bằng camera của thiết bị di động. Hình ảnh này được gửi đến backend, nơi mô hình Facenet được sử dụng để trích xuất vector đặc trưng (embedding) của khuôn mặt. Vector này sau đó được so sánh với các vector đã đăng ký trong cơ sở dữ liệu bằng một bộ phân loại (classifier) đã được huấn luyện để xác định danh tính và ghi nhận sự có mặt. Ứng dụng cũng cung cấp chức năng xem lại lịch sử điểm danh. Khóa luận tập trung vào các bước cần thiết để hoàn thành hệ thống, bao gồm phân tích yêu cầu, thiết kế kiến trúc, thiết kế cơ sở dữ liệu, xây dựng API backend, phát triển ứng dụng Flutter, tích hợp mô hình Facenet và đánh giá hiệu quả của hệ thống nhận diện.

***Từ khóa**: Nhận diện khuôn mặt, Hệ thống điểm danh, Facenet.*

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất tới TS. Ma Thị Châu – người đã tận tình chỉ bảo, góp ý và giúp đỡ em hoàn thành khóa luận tốt nghiệp.

Em xin được cảm ơn các thầy cô giáo Khoa Công Nghệ Thông Tin, trường Đại Học Công Nghệ - ĐHQGHN đã truyền đạt cho em những kiến thức vô cùng quý giá trong suốt quá trình học tập tại trường, tạo điều kiện tốt nhất cho em có thể hoàn thiện được khóa luận tốt nghiệp này.

Cuối cùng, em xin được cảm ơn những anh chị, bạn bè đã đồng hành giúp đỡ, động viên em trong suốt quá trình theo học tại trường.

Mặc dù đã rất cố gắng nhưng do kiến thức còn nhiều hạn chế nên khóa luận của em không thể tránh khỏi những sai sót. Em rất mong nhận được sự góp ý của các thầy cô và các bạn để em có thể hoàn thiện và khắc phục những thiếu sót của mình. Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi tên là Phạm Quý Dương sinh viên lớp ……Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Tôi xin cam đoan tất cả các kết quả trong tài liệu báo cáo này là do tôi tự tìm hiểu, tổng hợp và thực hiện. Tất cả các tài liệu đều có nguồn gốc rõ ràng và trích dẫn đầy đủ, không sao chép lại từ tổ chức hoặc cá nhân nào khác mà không chỉ rõ về mặt tài liệu tham khảo.

Tôi xin hoàn toàn chịu trách nhiệm với lời cam đoan trên. Nếu có gì sai trái, tôi xin chịu kỷ luật dưới mọi hình thức theo quy định.

Hà Nội, ngày ... tháng ... năm 2024

Sinh viên

Phạm Quý Dương

MỤC LỤC

BẢNG KÝ HIỆU, CHỮ VIẾT TẮT

STT	Ký hiệu, viết tắt	Ý nghĩa
1	CNN	Convolutional Neural Network
2	MTCNN	Multi-task Cascaded Convolutional Neural Network
3	API	Application Programming Interface
4	ORM	Object-Relational Mapping

Chương 1. MỞ ĐẦU

1.1. Giới thiệu chung

Trong bối cảnh thời đại công nghệ phát triển nhanh chóng, các công nghệ mới như Trí tuệ Nhân tạo và Sinh trắc học đang được phát triển và áp dụng rộng rãi vào nhiều lĩnh vực của cuộc sống. Đặc biệt, trong quản lý giáo dục và tổ chức, việc tự động hóa các quy trình thủ công ngày càng trở nên cấp thiết. Khi các giải pháp tự động được triển khai, lượng dữ liệu sinh ra, bao gồm thông tin người dùng, dữ liệu sinh trắc học, và các bản ghi hoạt động đòi hỏi phải được xử lý và lưu trữ một cách hiệu quả để đáp ứng những yêu cầu quan trọng như quản lý, thống kê, giám sát và đảm bảo an ninh.

Các dữ liệu này, đặc biệt là dữ liệu sinh trắc học và các bản ghi liên quan, đang trở thành một phần quan trọng của hệ thống thông tin quản lý hiện đại. Thế giới đang chứng kiến sự gia tăng trong việc ứng dụng AI và sinh trắc học để giải quyết các bài toán thực tế. Vì vậy, việc xử lý và quản lý hiệu quả nguồn dữ liệu này là hết sức quan trọng, đem lại nhiều lợi thế về hiệu quả và độ chính xác cho các trường học và tổ chức.

Trong lĩnh vực giáo dục, các trường học có thể tận dụng hệ thống điểm danh tự động để giảm thiểu thời gian và công sức quản lý, đồng thời tăng cường tính chính xác và minh bạch. Dữ liệu điểm danh chính xác giúp nhà trường và giáo viên theo dõi chuyên cần của sinh viên, đưa ra các biện pháp hỗ trợ kịp thời. Tương tự, trong môi trường doanh nghiệp, các hệ thống quản lý truy cập và chấm công dựa trên sinh trắc học giúp nâng cao an ninh và hiệu quả quản lý nhân sự.

Sự phát triển của AI và sinh trắc học không chỉ làm thay đổi cách thức quản lý và xác thực mà còn biến dữ liệu liên quan thành một tài sản quan trọng. Những tổ chức áp dụng hiệu quả các công nghệ này sẽ có ưu thế lớn trong việc tối ưu hóa hoạt động và nâng cao chất lượng quản lý. Điều này giải thích tại sao nhiều tổ chức đang tích cực đầu tư vào các giải pháp tự động hóa thông minh, nhằm cải thiện hiệu suất và độ tin cậy.

Khóa luận này trình bày Hệ thống Điểm danh Khuôn mặt sử dụng Facenet, một nền tảng ứng dụng công nghệ nhận diện khuôn mặt để tự động hóa quy trình điểm danh. Hệ thống bao gồm backend API, giao diện web cho quản lý, và ứng dụng di động cho người dùng cuối. Việc tách biệt các thành phần hệ thống, tối ưu hóa quy trình xử lý nhận diện (sử dụng Facenet và MTCNN (Multitask Cascaded Convolutional Networks)), và thiết kế cơ sở dữ liệu linh hoạt sẽ giúp hệ thống xử lý

hiệu quả, đảm bảo hoạt động ổn định, chính xác và có khả năng mở rộng khi quy mô người dùng và dữ liệu gia tăng.

1.2. Hệ thống điểm danh khuôn mặt sử dụng Facenet

Hệ thống Điểm danh Khuôn mặt sử dụng Facenet là nền tảng ứng dụng công nghệ nhận diện khuôn mặt để tự động hóa quy trình điểm danh trong môi trường giáo dục hoặc tổ chức. Nền tảng gồm hai thành phần chính: một ứng dụng web (Flask [3]) tối ưu hóa cho người quản lý (giáo viên) và một ứng dụng di động (Flutter [6]) cung cấp trải nghiệm điểm danh tiện lợi cho người dùng cuối (sinh viên).

Đặc điểm chính của hệ thống là khả năng nhận diện khuôn mặt để ghi nhận sự hiện diện. Sinh viên sử dụng ứng dụng di động để tham gia lớp học thông qua mã lớp, tải lên hình ảnh khuôn mặt của mình để hệ thống học và nhận diện. Khi cần điểm danh, sinh viên chỉ cần sử dụng camera trên ứng dụng di động để chụp ảnh, hệ thống sẽ tự động xử lý và ghi nhận trạng thái điểm danh. Sinh viên cũng có thể xem lại lịch sử điểm danh của mình ngay trên ứng dụng.

Đối với giáo viên, ứng dụng web cung cấp giao diện quản lý toàn diện. Giáo viên có thể tạo lớp học, thêm sinh viên vào lớp (kèm theo ảnh ban đầu nếu cần), và xem danh sách sinh viên cùng trạng thái điểm danh gần nhất. Hệ thống cũng hỗ trợ giáo viên thực hiện điểm danh cho lớp bằng cách tải lên một hình ảnh chứa nhiều khuôn mặt sinh viên, hệ thống sẽ xử lý ảnh và tự động nhận diện, cập nhật trạng thái cho các sinh viên có mặt. Giáo viên có thể xem lại chi tiết các bản ghi điểm danh theo ngày.

Về công nghệ, nền tảng sử dụng Flask cho backend API, SQLAlchemy để quản lý cơ sở dữ liệu, và ứng dụng các thư viện như MTCNN để phát hiện khuôn mặt và Facenet để trích xuất đặc trưng và so khớp nhận diện. Ứng dụng di động được xây dựng bằng Flutter [6], đảm bảo trải nghiệm đa nền tảng.

Hệ thống được thiết kế để xử lý yêu cầu từ cả web và mobile, lưu trữ dữ liệu người dùng, lớp học, hình ảnh và các bản ghi điểm danh. Hệ thống số hóa và tự động hóa quy trình điểm danh thủ công truyền thống. Nền tảng không chỉ là công cụ ghi nhận điểm danh, mà còn cung cấp một giải pháp quản lý tiện lợi cho giáo viên và trải nghiệm điểm danh nhanh chóng, hiện đại cho sinh viên, hướng tới mục tiêu tăng cường tính chính xác, tiết kiệm thời gian và tiện quản lý.

Chương 2. CƠ SỞ LÝ THUYẾT

Nội dung chính của chương này bao gồm hai mảng kiến thức trọng tâm. Thứ nhất, chương sẽ phân tích chi tiết về công nghệ nhận diện khuôn mặt, bắt đầu từ các thuật toán phát hiện khuôn mặt tiên tiến như MTCNN, các kỹ thuật tiền xử lý và căn chỉnh ảnh nhằm chuẩn hóa dữ liệu đầu vào, mô hình trích xuất đặc trưng sâu FaceNet dựa trên Triplet Loss , cho đến các phương pháp phân loại và so khớp hiệu quả như Độ tương đồng Cosine. Thứ hai, chương sẽ trình bày về các framework được lựa chọn để phát triển ứng dụng, bao gồm framework backend Flask phía máy chủ với các cơ chế định tuyến, xử lý yêu cầu, ORM , quản lý phiên và phiên bản cơ sở dữ liệu [4]; và framework Flutter phía client cho ứng dụng di động với kiến trúc giao diện, cơ chế giao tiếp backend,

2.1. Công nghệ nhận diện khuôn mặt

2.1.1. Các thuật toán phát hiện khuôn mặt

Phát hiện khuôn mặt (Face Detection) là bước đầu tiên và quan trọng trong quy trình nhận diện khuôn mặt tự động. Nhiệm vụ của nó là xác định vị trí và kích thước (thường là một hộp giới hạn - bounding box) của tất cả các khuôn mặt có trong một hình ảnh hoặc video. Đây là tiền đề cho các bước xử lý tiếp theo như căn chỉnh (alignment) và trích xuất đặc trưng (feature extraction). Việc phát hiện chính xác khuôn mặt, ngay cả trong các điều kiện phức tạp (ánh sáng thay đổi, góc nhìn khác nhau, bị che khuất một phần), là yếu tố then chốt ảnh hưởng đến hiệu suất tổng thể của hệ thống nhận diện.

Hệ thống điểm danh này sử dụng MTCNN [7], một thuật toán học sâu hiệu quả và phổ biến cho việc phát hiện và căn chỉnh khuôn mặt. MTCNN nổi bật với kiến trúc tầng (cascaded) gồm ba mạng Nơ-ron Tích chập (CNN) hoạt động nối tiếp, đồng thời thực hiện nhiều nhiệm vụ (multi-task).

MTCNN xử lý ảnh đầu vào qua ba giai đoạn mạng CNN, giúp tăng dần độ chính xác và loại bỏ các vùng không phải khuôn mặt một cách hiệu quả. Các mạng này bao gồm: P-Net (Proposal Network) quét nhanh qua ảnh đầu vào ở nhiều tỷ lệ khác nhau (thông qua một kim tự tháp ảnh) để đề xuất các vùng có khả năng chứa khuôn mặt và ước lượng sơ bộ hộp giới hạn. Các vùng có độ tin cậy thấp sẽ bị loại bỏ sớm. R-Net (Refinement Network) nhận các vùng ứng viên từ P-Net, thực hiện phân loại chi tiết hơn để loại bỏ các ứng viên sai và tinh chỉnh lại vị trí hộp giới hạn chính xác hơn.O-Net (Output Network) nhận các ứng viên còn lại từ R-Net, thực hiện phân loại

lần cuối, tinh chỉnh hộp giới hạn một cách chính xác nhất và đồng thời dự đoán vị trí của các điểm mốc quan trọng trên khuôn mặt, thường là 5 điểm: hai mắt, mũi và hai khóe miệng.

Song song với kiến trúc tầng là chiến lược học đa nhiệm, nơi hai mạng R-Net và O-Net được huấn luyện đồng thời ba tác vụ khác nhau: phân loại (cls), hồi quy hộp giới hạn (box) và định vị điểm mốc (landmark). Hàm mất mát tổng hợp được biểu diễn bởi:

$$L = L_{\text{cls}} + \alpha\, L_{\text{box}} + \beta\, L_{\text{landmark}} \tag{1}$$

Trong đó:

$$L_{\text{cls}} = -\big[y \log p + (1-y)\log(1-p)\big], \tag{2}$$

$$L_{\text{box}} = \|\mathbf{b}_{\text{pred}} - \mathbf{b}_{\text{true}}\|^2, \tag{3}$$

$$L_{\text{landmark}} = \|\mathbf{l}_{\text{pred}} - \mathbf{l}_{\text{true}}\|^2. \tag{4}$$

Các hệ số α và β được điều chỉnh để cân bằng tầm quan trọng giữa việc phân loại, tinh chỉnh hộp và dự đoán điểm mốc.

2.1.2. Kỹ thuật tiền xử lý và căn chỉnh khuôn mặt

Trong hệ thống điểm danh, một khi MTCNN đã phát hiện khuôn mặt và trả về tọa độ hộp giới hạn, bước tiếp theo là tiền xử lý và căn chỉnh để đảm bảo đầu vào cho FaceNet luôn đồng nhất. Đầu tiên, hình ảnh gốc được cắt theo tọa độ hộp giới hạn kèm lề 20% so với cạnh nhỏ nhất. Việc thêm lề này không chỉ giữ nguyên toàn bộ vùng mặt mà còn bao gồm một phần ngữ cảnh xung quanh, giúp giảm hiện tượng cắt khung hình quá sát và hỗ trợ độ ổn định khi căn chỉnh.

Tiếp theo, khung hình đã cắt được thay đổi kích thước về 160x160 pixel, kích thước tiêu chuẩn mà FaceNet yêu cầu. Phương pháp nội suy được sử dụng là cv2.INTER_CUBIC[2], vốn duy trì chi tiết hình ảnh tốt hơn khi so sánh với các phương pháp nội suy khác. Sau khi thu được ảnh đúng kích thước, toàn bộ giá trị pixel nguyên thủy (trong khoảng 0–255) được chuẩn hóa về phạm vi [0,1] bằng cách chia cho 255.0 và chuyển sang kiểu float32. Bước này giúp dữ liệu đồng quy về cùng một thang đo, giảm thiểu sự khác biệt do điều kiện ánh sáng và độ sáng từng ảnh.

Cuối cùng, kỹ thuật prewhitening được áp dụng để làm trắng ảnh, theo công

thức:

$$\text{whitened\_img} = \frac{\text{img} - \mu}{\sigma_{\text{adj}}} \tag{5}$$

Trong đó $\mu = \text{mean}(\text{img})$ là giá trị trung bình và $\sigma_{\text{adj}} = \max\left(\text{std}(\text{img}), \frac{1}{\sqrt{N}}\right)$ là độ lệch chuẩn đã điều chỉnh, với N là số điểm ảnh. Việc đảm bảo mẫu số không quá nhỏ tránh chia cho giá trị xấp xỉ 0, đồng thời đẩy phương sai về 1. Kết quả của prewhitening là hình ảnh có phân phối giá trị xung quanh 0, giúp tăng độ tương phản các đặc trưng và giảm thiểu ảnh hưởng của ánh sáng.

2.1.3. Mô hình trích xuất đặc trưng khuôn mặt

Sau khi khuôn mặt được phát hiện và bước tiền xử lý đã chuẩn hóa kích thước, giá trị pixel và căn chỉnh cơ bản, mạng FaceNet [6] bắt đầu quá trình biến đổi ảnh khuôn mặt thành vector embedding. Ảnh khuôn mặt đã qua prewhitening (160×160×3) được đưa vào kiến trúc CNN nền tảng, thường là các biến thể Inception, để trích xuất các đặc trưng sâu. Lớp cuối cùng của mạng, trước khi lớp phân loại nếu có, cho ra một vector embedding với kích thước cố định (thường là 128 hoặc 512 chiều). Quá trình này đảm bảo các embedding nắm bắt bản chất nhận dạng của khuôn mặt sao cho các ảnh của cùng một người tạo ra các vector gần nhau, trong khi các người khác nhau tạo ra các vector cách xa nhau.

FaceNet được huấn luyện sử dụng Triplet Loss , được định nghĩa bởi công thức:

$$L = \sum_{i=1}^{N} \left[\|f(A_i) - f(P_i)\|_2^2 \; - \; \|f(A_i) - f(N_i)\|_2^2 \; + \; \alpha \right]_+ \tag{6}$$

Trong đó mỗi bộ triplet (A_i, P_i, N_i) gồm ảnh Anchor, Positive (cùng người với Anchor) và Negative (khác người), $/alpha$ là ngưỡng khoảng cách tối thiểu, và $[z]_+ = \max(z, 0)$ đảm bảo hàm mất mát chỉ dương khi điều kiện không thỏa mãn.

Cuối cùng, sau khi lấy được vector embedding thô từ mô hình, ta thực thi chuẩn hóa L2:

$$\mathbf{e}_{\text{norm}} = \frac{\mathbf{e}}{\|\mathbf{e}\|_2} \tag{7}$$

Việc này giữ cho tất cả embedding nằm trên mặt cầu đơn vị cho việc so khớp bằng Cosine Similarity—phương thức chính được dùng trong hệ thống điểm danh.

2.1.4. Phương pháp phân loại và so khớp khuôn mặt

Sau khi embedding của một khuôn mặt mới được tạo ra từ Facenet và chuẩn hóa L2, bước cuối cùng trong quy trình nhận diện khuôn mặt là so khớp embedding này với tập hợp các embedding đã lưu của sinh viên trong cơ sở dữ liệu. Mục tiêu của việc so khớp là xác định xem một embedding mới có đủ "gần" với một embedding đã biết để được coi là cùng một người hay không.

Phương pháp được hệ thống sử dụng là Độ tương đồng Cosine[5], vốn đo lường góc giữa hai vector trong không gian tích vô hướng mà không phụ thuộc vào độ lớn của chúng. Công thức chung để tính độ tương đồng Cosine giữa hai vector A và B là:

$$\text{Similarity}(A, B) = \frac{A \cdot B}{\|A\|_2 \, \|B\|_2} \tag{8}$$

Khi cả hai vector đã được chuẩn hóa L2 (tức$\|A\|_2 = \|B\|_2 = 1$), biểu thức này đơn giản thành:

$$\text{Similarity}(A, B) = A \cdot B \tag{9}$$

tức tích vô hướng giữa hai embedding.

Quy trình so khớp diễn ra như sau: với mỗi embedding đã lưu, ta tính độ tương đồng Cosine với embedding mới, rồi so sánh kết quả đó với một ngưỡng đã định trước là 0.6. Nếu độ tương đồng lớn hơn ngưỡng và đồng thời là giá trị lớn nhất trong tất cả các embedding đối chiếu, ta kết luận là có "match" và trả về danh tính tương ứng; ngược lại, hệ thống sẽ xác định không có cá nhân nào khớp.

2.2. Framework phát triển ứng dụng web phía máy chủ

2.2.1. Cơ chế định tuyến và xử lý yêu cầu

Hoạt động cốt lõi của backend trong hệ thống điểm danh này dựa trên cơ chế định tuyến và xử lý yêu cầu do framework Flask cung cấp. Định tuyến là quá trình máy chủ xác định logic xử lý nào sẽ được thực thi khi nhận được một yêu cầu từ client, dựa trên URL đích và phương thức HTTP (GET hoặc POST) của yêu cầu đó. Flask sử dụng các decorator trong Python để liên kết các đường dẫn URL với các hàm xử lý cụ thể. Để tăng tính tổ chức, hệ thống nhóm các route liên quan vào các module riêng biệt bằng cách sử dụng Blueprints. Mỗi Blueprint đại diện cho một tập hợp chức

năng (ví dụ: xác thực, API cho di động) và được đăng ký với ứng dụng chính, đi kèm một tiền tố URL để tạo không gian tên cho các đường dẫn thuộc về nó.

Khi một yêu cầu đến khớp với một route đã đăng ký, Flask kích hoạt hàm xử lý tương ứng và bắt đầu quá trình xử lý yêu cầu. Hàm này có quyền truy cập vào dữ liệu của yêu cầu đến thông qua một đối tượng đặc biệt do Flask cung cấp. Hệ thống tận dụng đối tượng này để trích xuất thông tin từ nhiều nguồn khác nhau: dữ liệu gửi từ các form HTML trong giao diện web, các file ảnh được tải lên, các tham số được truyền qua day du lieu của URL trong các lệnh gọi API, và dữ liệu có cấu trúc JSON gửi trong phần thân của các yêu cầu API từ ứng dụng di động.

Với dữ liệu yêu cầu đã được trích xuất, hàm xử lý thực hiện logic nghiệp vụ cần thiết. Điều này bao gồm việc tương tác với cơ sở dữ liệu để truy vấn hoặc cập nhật thông tin (thông qua các định nghĩa model), gọi đến các module tiện ích để thực hiện các tác vụ chuyên biệt như xử lý ảnh và nhận diện khuôn mặt, và sử dụng các cơ chế tích hợp của framework để quản lý trạng thái đăng nhập và phiên làm việc của người dùng.

Sau khi hoàn thành logic xử lý, hàm phải tạo và trả về một phản hồi HTTP cho client. Hệ thống này tạo ra nhiều loại phản hồi khác nhau tùy thuộc vào ngữ cảnh. Đối với giao diện web, nó tạo ra các trang HTML động bằng cách sử dụng một template engine tích hợp (Jinja2), điền dữ liệu vào các mẫu định sẵn. Đối với các yêu cầu API từ ứng dụng di động, nó tuần tự hóa dữ liệu kết quả thành định dạng JSON. Trong các luồng hoạt động web, hệ thống cũng sử dụng các phản hồi chuyển hướng HTTP để điều hướng trình duyệt của người dùng đến một URL khác sau khi một hành động hoàn tất, đồng thời cung cấp cơ chế để hiển thị các thông báo ngắn (flash messages) trên trang kết quả.

2.2.2. Hệ thống template Engine

Để tạo ra các trang web động cho giao diện người dùng phía giáo viên, hệ thống backend sử dụng một Template Engine. Vai trò của nó là kết hợp logic xử lý dữ liệu từ backend với cấu trúc trình bày HTML, cho phép tạo ra các trang web có nội dung thay đổi dựa trên dữ liệu cụ thể của từng yêu cầu. Hệ thống này sử dụng Jinja2, một template engine tích hợp chặt chẽ với Flask. Jinja2 xử lý các file mẫu (templates), là các file HTML chứa trong thư mục templates có bổ sung các cú pháp đặc biệt. Khi backend cần hiển thị một trang web, nó chuẩn bị dữ liệu và chỉ định file template cần dùng. Jinja2 sau đó đọc template, thay thế các cú pháp đặc biệt bằng dữ liệu được cung

cấp và thực thi các chỉ thị logic để tạo ra file HTML cuối cùng gửi đến trình duyệt.

Trong các file template của hệ thống, Jinja2 cho phép nhúng trực tiếp giá trị của các biến từ backend vào HTML bằng cách sử dụng cú pháp dấu ngoặc nhọn kép ({{ variable_name }}). Nó cũng cung cấp các cấu trúc điều khiển, nằm trong dấu ngoặc nhọn và phần trăm ({% ... %}), để thực hiện logic trình bày. Hệ thống sử dụng các cấu trúc điều kiện (if/else/endif) để hiển thị hoặc ẩn các phần HTML tùy thuộc vào dữ liệu, và các vòng lặp (for/endfor) để duyệt qua các tập hợp dữ liệu (ví dụ: danh sách sinh viên, lớp học) và tạo ra các đoạn mã HTML lặp lại tương ứng.

Một tính năng quan trọng khác của Jinja2 được hệ thống tận dụng là kế thừa template. Một file template cơ sở (base.html) định nghĩa cấu trúc HTML chung cho toàn bộ trang web, bao gồm các vùng nội dung có thể tùy chỉnh được đánh dấu bằng các khối (block). Các template cụ thể khác (ví dụ: trang dashboard, trang đăng nhập) kế thừa từ template cơ sở này và chỉ cần định nghĩa lại nội dung cho các khối tương ứng. Cơ chế này giúp tái sử dụng mã HTML chung, đảm bảo tính nhất quán về giao diện và đơn giản hóa việc bảo trì. Ngoài ra, các template còn sử dụng các hàm tiện ích được cung cấp bởi Flask và Jinja2, chẳng hạn như hàm để tạo URL động đến các route khác trong ứng dụng và hàm để truy cập các thông báo ngắn được gửi từ backend.

2.2.3. Object-Relational Mapping

Trong các ứng dụng web tương tác với cơ sở dữ liệu quan hệ, việc chuyển đổi dữ liệu giữa mô hình đối tượng trong mã nguồn (ví dụ: các lớp Python) và mô hình bảng trong cơ sở dữ liệu (ví dụ: các bảng SQL) là một tác vụ phổ biến và lặp đi lặp lại. Kỹ thuật Object-Relational Mapping (ORM) ra đời để tự động hóa và đơn giản hóa quá trình này. ORM hoạt động như một lớp trừu tượng, cho phép lập trình viên tương tác với cơ sở dữ liệu thông qua các đối tượng và phương thức của ngôn ngữ lập trình đang sử dụng, thay vì phải viết các câu lệnh SQL thủ công. Hệ thống backend của ứng dụng điểm danh này sử dụng SQLAlchemy, một thư viện ORM mạnh mẽ và linh hoạt cho Python, được tích hợp thông qua phần mở rộng Flask-SQLAlchemy.

Với SQLAlchemy, mỗi bảng trong cơ sở dữ liệu được ánh xạ tới một lớp Python, thường được gọi là "model" (được định nghĩa trong file models.py của hệ thống). Các lớp model này kế thừa từ một lớp cơ sở do SQLAlchemy cung cấp (trong trường hợp Flask-SQLAlchemy là db.Model), và các thuộc tính của lớp tương ứng với các cột trong bảng cơ sở dữ liệu. SQLAlchemy tự động quản lý việc ánh xạ này, bao gồm cả kiểu dữ liệu giữa Python và SQL. Điều này cho phép lập trình viên làm việc với dữ

liệu dưới dạng các đối tượng Python quen thuộc.

SQLAlchemy quản lý các tương tác với cơ sở dữ liệu thông qua khái niệm "Session". Session hoạt động như một khu vực trung gian (unit of work) nơi các thay đổi trên đối tượng model được theo dõi. Khi lập trình viên tạo, sửa đổi hoặc xóa các đối tượng model, những thay đổi này được ghi nhận trong Session. Sau đó, khi Session được "commit", SQLAlchemy sẽ tự động tạo ra các câu lệnh SQL tương ứng (INSERT, UPDATE, DELETE) và thực thi chúng trên cơ sở dữ liệu trong một giao dịch (transaction). Flask-SQLAlchemy quản lý vòng đời của Session một cách tự động trong ngữ cảnh của một yêu cầu web.

Một trong những lợi ích chính của việc sử dụng ORM như SQLAlchemy là khả năng truy vấn dữ liệu bằng cách sử dụng cú pháp Python thay vì SQL. SQLAlchemy cung cấp một API truy vấn phong phú, cho phép xây dựng các truy vấn phức tạp thông qua việc gọi các phương thức trên các đối tượng model và session. Các thao tác như lọc (filtering), sắp xếp (ordering), kết nối bảng (joining), và tổng hợp dữ liệu (aggregation) đều có thể được thực hiện bằng các biểu thức Python, giúp mã nguồn trở nên dễ đọc và gần gũi hơn với logic nghiệp vụ. SQLAlchemy sẽ dịch các truy vấn này thành các câu lệnh SQL tối ưu cho hệ quản trị cơ sở dữ liệu cụ thể đang được sử dụng.

Ngoài ra, SQLAlchemy hỗ trợ định nghĩa và quản lý các mối quan hệ giữa các bảng/model khác nhau, chẳng hạn như quan hệ một-nhiều (one-to-many) hoặc nhiều-nhiều (many-to-many), điều này rất quan trọng trong việc mô hình hóa cấu trúc dữ liệu phức tạp của ứng dụng. Các mối quan hệ này được khai báo trong các lớp model, cho phép dễ dàng truy cập dữ liệu liên quan thông qua các thuộc tính của đối tượng (ví dụ: truy cập danh sách sinh viên thuộc về một lớp học).

Việc sử dụng SQLAlchemy ORM trong hệ thống điểm danh này mang lại sự trừu tượng hóa cao, giúp giảm đáng kể lượng mã SQL cần viết và quản lý, đồng thời tăng năng suất phát triển và làm cho mã nguồn dễ bảo trì hơn. Nó cũng cung cấp một mức độ độc lập nhất định với hệ quản trị cơ sở dữ liệu cụ thể, mặc dù các tính năng nâng cao có thể vẫn phụ thuộc vào CSDL.

2.2.4. Cơ chế xác thực và quản lý phiên

Trong một ứng dụng đa người dùng như hệ thống điểm danh, việc xác định danh tính người dùng (Xác thực - Authentication) và duy trì trạng thái đăng nhập của họ qua nhiều yêu cầu khác nhau (Quản lý Phiên - Session Management) là cực kỳ quan trọng. Backend của hệ thống này sử dụng phần mở rộng Flask-Login để xử lý các tác vụ này

một cách an toàn và hiệu quả. Flask-Login cung cấp một khung làm việc để quản lý các khía cạnh phức tạp của việc đăng nhập, đăng xuất và duy trì phiên làm việc của người dùng.

Quá trình xác thực bắt đầu khi người dùng (giáo viên qua giao diện web hoặc sinh viên qua ứng dụng di động) cung cấp thông tin định danh, điển hình là tên người dùng (hoặc mã số) và mật khẩu. Backend nhận thông tin này và so sánh nó với dữ liệu được lưu trữ trong cơ sở dữ liệu. Để đảm bảo an toàn, hệ thống không lưu trữ mật khẩu dưới dạng văn bản gốc mà sử dụng các hàm băm mật mã (cryptographic hashing) để tạo ra một chuỗi đại diện không thể đảo ngược từ mật khẩu. Khi người dùng đăng nhập, mật khẩu họ cung cấp sẽ được băm và so sánh với chuỗi băm đã lưu. Nếu khớp, danh tính người dùng được xác nhận.

Sau khi xác thực thành công, Flask-Login thiết lập một phiên làm việc (session) cho người dùng. Cơ chế này hoạt động bằng cách lưu trữ một định danh duy nhất và an toàn cho phiên người dùng trong một cookie được ký điện tử (signed cookie) trên trình duyệt của người dùng (hoặc được quản lý bởi ứng dụng di động). Cookie này được gửi kèm theo mỗi yêu cầu tiếp theo từ client đến server. Flask-Login tự động giải mã cookie này, xác định người dùng tương ứng và tải thông tin của họ (từ model User hoặc Student) vào một đối tượng đặc biệt, giúp backend dễ dàng truy cập thông tin người dùng đang đăng nhập trong suốt quá trình xử lý yêu cầu.

Flask-Login cũng cung cấp các cơ chế để kiểm soát quyền truy cập (Authorization). Hệ thống sử dụng các decorator được cung cấp bởi Flask-Login để bảo vệ các route hoặc chức năng nhất định. Khi một yêu cầu đến một route được bảo vệ, decorator này sẽ kiểm tra xem người dùng hiện tại đã được xác thực hay chưa (thông qua việc kiểm tra phiên làm việc). Nếu người dùng chưa đăng nhập, họ sẽ bị từ chối truy cập, thường là bằng cách chuyển hướng đến trang đăng nhập (đối với web) hoặc trả về lỗi (đối với API). Điều này đảm bảo rằng chỉ những người dùng đã xác thực mới có thể thực hiện các hành động hoặc truy cập các tài nguyên được bảo vệ. Quá trình đăng xuất sẽ xóa thông tin phiên khỏi cookie, kết thúc phiên làm việc của người dùng.

2.2.5. Mô hình dữ liệu

Để lưu trữ và quản lý thông tin một cách có cấu trúc, hệ thống điểm danh dựa trên một mô hình dữ liệu được định nghĩa rõ ràng trong cơ sở dữ liệu quan hệ. Mô hình dữ liệu này xác định các thực thể chính mà hệ thống cần quản lý, các thuộc tính

của từng thực thể, và các mối quan hệ giữa chúng. Trong backend Flask, mô hình dữ liệu này được hiện thực hóa thông qua các lớp Python, hay còn gọi là các "model", bằng cách sử dụng SQLAlchemy ORM

Các lớp model này ánh xạ trực tiếp tới các bảng trong cơ sở dữ liệu. Mỗi lớp đại diện cho một bảng, và các thuộc tính được khai báo bên trong lớp (sử dụng các cấu trúc của SQLAlchemy như db.Column) tương ứng với các cột của bảng đó, bao gồm cả việc định nghĩa kiểu dữ liệu (ví dụ: chuỗi, số nguyên, ngày giờ) và các ràng buộc (constraints) như khóa chính (primary key), khóa ngoại (foreign key), tính duy nhất (unique), hay giá trị mặc định. Các thực thể chính được mô hình hóa trong hệ thống bao gồm người dùng (giáo viên), sinh viên, lớp học, và các bản ghi điểm danh.

Các mối quan hệ giữa các thực thể này cũng được định nghĩa trong các lớp model bằng cách sử dụng các hàm quan hệ của SQLAlchemy). Ví dụ, mối quan hệ một-nhiều (one-to-many) được thiết lập giữa một lớp học và các sinh viên thuộc lớp đó, hoặc giữa một lớp học và các buổi điểm danh của lớp đó. Mối quan hệ nhiều-nhiều (many-to-many) được sử dụng để liên kết sinh viên với các lớp học mà họ tham gia, thường thông qua một bảng trung gian. Việc định nghĩa các mối quan hệ này không chỉ thiết lập các ràng buộc khóa ngoại trong cơ sở dữ liệu mà còn cho phép ORM cung cấp cách thức thuận tiện để truy cập dữ liệu liên quan thông qua các thuộc tính trên đối tượng model trong mã Python.

Thông qua các lớp model này, SQLAlchemy có được "bản thiết kế" chi tiết của cấu trúc cơ sở dữ liệu, cho phép nó thực hiện các thao tác như tạo bảng , truy vấn dữ liệu, và quản lý các mối quan hệ một cách hiệu quả, đồng thời cung cấp một giao diện lập trình hướng đối tượng để tương tác với dữ liệu trong backend.

2.2.6. Quản lý phiên bản cơ sở dữ liệu

Trong quá trình phát triển ứng dụng, cấu trúc của cơ sở dữ liệu (schema) - bao gồm các bảng, cột, và mối quan hệ - thường xuyên thay đổi để đáp ứng các yêu cầu mới hoặc điều chỉnh logic nghiệp vụ. Việc quản lý những thay đổi này một cách thủ công, đặc biệt là khi ứng dụng đã được triển khai, rất dễ xảy ra lỗi và thiếu nhất quán. Để giải quyết vấn đề này, hệ thống backend sử dụng cơ chế Quản lý Phiên bản Cơ sở dữ liệu (Database Migrations) thông qua thư viện Alembic, được tích hợp với Flask bằng phần mở rộng Flask-Migrate.

Alembic cho phép theo dõi các thay đổi trong định nghĩa model SQLAlchemy và tự động tạo ra các kịch bản di chuyển dữ liệu. Mỗi kịch bản này chứa các lệnh cần

thiết để cập nhật cấu trúc cơ sở dữ liệu từ một phiên bản cũ sang một phiên bản mới .

Khi có sự thay đổi trong các lớp model, Flask-Migrate cung cấp các lệnh để so sánh trạng thái hiện tại của mô hình với trạng thái cuối cùng được ghi nhận trong cơ sở dữ liệu (thông qua một bảng đặc biệt do Alembic quản lý). Dựa trên sự khác biệt này, một kịch bản di chuyển mới sẽ được tự động tạo ra. Kịch bản này chứa mã Python sử dụng API của Alembic để thực hiện các thay đổi schema cần thiết.

Để áp dụng các thay đổi lên cơ sở dữ liệu thực tế (dù là môi trường phát triển hay sản xuất), một lệnh khác của Flask-Migrate sẽ được thực thi. Lệnh này sẽ chạy các kịch bản di chuyển chưa được áp dụng theo đúng thứ tự phiên bản, đảm bảo cơ sở dữ liệu được cập nhật lên cấu trúc mới nhất tương ứng với mã nguồn. Alembic cũng hỗ trợ việc hạ cấp (downgrade) cơ sở dữ liệu về các phiên bản cũ hơn nếu cần thiết, bằng cách thực thi logic ngược lại được định nghĩa trong các kịch bản di chuyển. Việc sử dụng Alembic và Flask-Migrate giúp tự động hóa quy trình cập nhật schema, đảm bảo tính nhất quán và giảm thiểu rủi ro khi thay đổi cấu trúc cơ sở dữ liệu trong suốt vòng đời của ứng dụng.

2.3. Framework phát triển ứng dụng di động

2.3.1. Kiến trúc và thành phần giao diện người dùng

Ứng dụng di động dành cho sinh viên trong hệ thống điểm danh này được xây dựng bằng cách sử dụng Flutter, một bộ công cụ giao diện người dùng mã nguồn mở do Google phát triển. Flutter cho phép tạo ra các ứng dụng được biên dịch trực tiếp sang mã máy cho nhiều nền tảng, bao gồm Android và iOS, từ một cơ sở mã nguồn duy nhất viết bằng ngôn ngữ lập trình Dart. Việc lựa chọn Flutter giúp đảm bảo trải nghiệm người dùng nhất quán trên các thiết bị khác nhau và tối ưu hóa quá trình phát triển cũng như bảo trì ứng dụng.

Nguyên tắc cốt lõi của Flutter là xây dựng giao diện người dùng hoàn toàn từ các thành phần gọi là Widgets. Mọi thứ hiển thị trên màn hình, từ các yếu tố cấu trúc cơ bản như bố cục, khoảng đệm đến các thành phần tương tác phức tạp như nút bấm, trường nhập liệu, và danh sách cuộn, đều là các Widget. Flutter cung cấp một bộ sưu tập phong phú các Widget được thiết kế sẵn, đồng thời cho phép tạo ra các Widget tùy chỉnh. Giao diện người dùng được xây dựng theo cách khai báo, nghĩa là lập trình viên mô tả trạng thái mong muốn của UI, và Flutter sẽ tự động cập nhật giao diện khi trạng thái đó thay đổi.

Để quản lý dữ liệu và sự thay đổi của nó ảnh hưởng đến giao diện người dùng,

Flutter sử dụng các cơ chế Quản lý Trạng thái . Khi dữ liệu trong ứng dụng thay đổi, cơ chế quản lý trạng thái sẽ thông báo cho các Widget liên quan để chúng tự động cập nhật lại giao diện, phản ánh đúng dữ liệu mới nhất. Hệ thống này áp dụng các kỹ thuật quản lý trạng thái phù hợp để xử lý luồng dữ liệu giữa các màn hình và các thành phần giao diện khác nhau.

Thông qua Flutter, ứng dụng di động cung cấp các chức năng cần thiết cho sinh viên, bao gồm đăng nhập, đăng ký khuôn mặt, tham gia lớp học, thực hiện điểm danh và xem lại lịch sử điểm danh. Toàn bộ logic giao diện và tương tác người dùng được triển khai bằng Dart, giao tiếp với backend thông qua các API đã được định nghĩa để trao đổi dữ liệu.

2.3.2. Giao tiếp Backend

Trong kiến trúc ứng dụng di động hiện đại, giao tiếp với backend đóng vai trò then chốt, đặc biệt đối với hệ thống điểm danh đòi hỏi trao đổi dữ liệu phức tạp như ảnh khuôn mặt. Hệ thống điểm danh này áp dụng mô hình client-server tiêu chuẩn, trong đó ứng dụng Flutter (client) giao tiếp với API của Flask (server) thông qua giao thức HTTP.

Về phương thức truyền tải, hệ thống sử dụng các yêu cầu HTTP với nhiều phương thức khác nhau: GET cho việc truy xuất thông tin (như lịch sử điểm danh), POST cho việc gửi dữ liệu mới (như đăng ký, điểm danh). Dữ liệu được trao đổi chủ yếu ở định dạng JSON, được tuần tự hóa và giải tuần tự hóa giữa các đối tượng Dart và chuỗi JSON. Đây là một cách tiếp cận phổ biến vì tính nhẹ nhàng và khả năng tương thích cao của JSON trên nhiều nền tảng.

Một khía cạnh quan trọng của giao tiếp API là xử lý bất đồng bộ. Flutter xử lý các hoạt động mạng trong một cách không đồng bộ để không chặn luồng chính của ứng dụng, đảm bảo giao diện người dùng vẫn phản hồi trong khi đợi kết quả từ server. Điều này được thực hiện thông qua mô hình lập trình bất đồng bộ với Future và async/await, cho phép viết mã xử lý không đồng bộ theo cách tuần tự và dễ hiểu.

Xác thực là một khía cạnh quan trọng khác trong giao tiếp client-server. Hệ thống sử dụng cơ chế xác thực dựa trên ID, trong đó một định danh duy nhất được lưu trữ cục bộ sau khi đăng nhập thành công và được gửi kèm với mỗi yêu cầu API tiếp theo. Phương pháp này cân bằng giữa sự đơn giản và bảo mật, phù hợp với quy mô của hệ thống.

Đối với việc truyền tải dữ liệu nhị phân như hình ảnh, hệ thống sử dụng các yêu

cầu HTTP multipart/form-data, cho phép gửi cả dữ liệu văn bản (như ID sinh viên, thông tin lớp học) và tệp tin trong cùng một yêu cầu. Phương pháp này đặc biệt hiệu quả cho các tác vụ như tải lên nhiều ảnh khuôn mặt hoặc gửi ảnh selfie để điểm danh.

2.3.3. Quản lý trạng thái

Quản lý trạng thái là một trong những thách thức cốt lõi trong phát triển ứng dụng di động, đặc biệt là với các ứng dụng có nhiều màn hình và luồng dữ liệu phức tạp như hệ thống điểm danh. Trong Flutter, quản lý trạng thái liên quan đến việc duy trì và cập nhật dữ liệu ảnh hưởng đến giao diện người dùng, đảm bảo UI phản ánh chính xác trạng thái hiện tại của ứng dụng.

Kiến trúc quản lý trạng thái của ứng dụng áp dụng mô hình hỗn hợp, kết hợp các phương pháp khác nhau tùy thuộc vào phạm vi và độ phức tạp của dữ liệu. Đối với trạng thái nội bộ của một widget, hệ thống dựa vào StatefulWidget của Flutter, cho phép các component duy trì và cập nhật trạng thái của riêng mình. Khi trạng thái thay đổi, Flutter tự động xây dựng lại UI để phản ánh các thay đổi, tạo ra một vòng đời của widget có thể dự đoán và quản lý được.

Đối với trạng thái phiên làm việc và dữ liệu người dùng kéo dài, ứng dụng sử dụng lưu trữ cục bộ. Đây là một phương pháp đơn giản nhưng hiệu quả, lưu trữ thông tin như ID người dùng hoặc cài đặt ưu tiên trong bộ nhớ không bay hơi của thiết bị. Dữ liệu này tồn tại ngay cả khi ứng dụng bị đóng, cho phép duy trì trạng thái đăng nhập hoặc cài đặt người dùng giữa các phiên sử dụng.

Quan trọng không kém là việc truyền trạng thái giữa các màn hình. Ứng dụng áp dụng cơ chế truyền dữ liệu thông qua điều hướng, nơi các tham số có thể được gửi khi một màn hình mới được đưa vào ngăn xếp điều hướng. Điều này không chỉ đảm bảo dữ liệu sẵn có ngay lập tức cho màn hình mới mà còn giảm thiểu nhu cầu truy vấn lại thông tin từ server. Hệ thống cũng triển khai mẫu giao diện người dùng theo trạng thái, trong đó UI thay đổi dựa trên trạng thái hiện tại của dữ liệu hoặc quá trình. Ví dụ, màn hình có thể hiển thị chỉ báo tải trong khi đang chờ phản hồi từ API, nội dung chính khi dữ liệu có sẵn, hoặc thông báo lỗi nếu yêu cầu thất bại. Cơ chế này cung cấp phản hồi trực quan cho người dùng và đảm bảo tính nhất quán của trải nghiệm.

2.3.4. Xử lý ảnh phía Client

Xử lý ảnh ở phía client là một thành phần thiết yếu của hệ thống điểm danh dựa trên nhận diện khuôn mặt, liên quan đến việc thu thập, xử lý và truyền dữ liệu hình ảnh. Quy trình này cân bằng giữa trải nghiệm người dùng, hiệu suất thiết bị và tải trọng mạng.

Quá trình bắt đầu với việc thu thập hình ảnh, nơi ứng dụng tương tác với API camera của thiết bị di động. Đây là bước quan trọng để đảm bảo chất lượng ảnh đầu vào cho hệ thống nhận diện khuôn mặt. Ứng dụng hỗ trợ hai phương thức thu thập: chụp ảnh trực tiếp thông qua camera của thiết bị, cung cấp hình ảnh thời gian thực cho việc điểm danh, và chọn ảnh từ thư viện phương tiện, thuận tiện cho việc đăng ký nhiều ảnh khuôn mặt ban đầu.

Sau khi thu thập, hình ảnh trải qua một quy trình quản lý cục bộ trước khi được truyền lên server. Ảnh được lưu trữ tạm thời trong bộ nhớ của thiết bị với định danh duy nhất để tham chiếu trong quá trình xử lý tiếp theo. Trong một số trường hợp, các thao tác tiền xử lý cơ bản có thể được áp dụng trực tiếp trên thiết bị, chẳng hạn như điều chỉnh kích thước để giảm kích thước tệp truyền tải mà vẫn duy trì đủ chất lượng cho nhận diện khuôn mặt.

Giao diện người dùng cung cấp khả năng xem trước và xác nhận ảnh, cho phép người dùng đánh giá chất lượng hình ảnh trước khi gửi lên server. Đây là một bước quan trọng để đảm bảo ảnh đủ rõ nét, được chiếu sáng đầy đủ và khuôn mặt được đặt đúng vị trí, tất cả đều là yếu tố then chốt để đảm bảo độ chính xác của nhận diện.

Khi truyền tải ảnh, ứng dụng sử dụng cơ chế hiệu quả để gửi dữ liệu nhị phân lớn qua HTTP. Ảnh được mã hóa thành định dạng phù hợp và truyền như một phần của yêu cầu HTTP multipart, cùng với các siêu dữ liệu cần thiết như định danh người dùng và thông tin lớp học. Đối với việc tải lên hàng loạt, như trong quá trình đăng ký khuôn mặt, nhiều ảnh được xử lý tuần tự trong một yêu cầu duy nhất.

Cách tiếp cận này tận dụng hiệu quả khả năng của thiết bị di động để thu thập hình ảnh chất lượng cao, trong khi chuyển phần lớn khối lượng tính toán nặng nề của việc nhận diện khuôn mặt sang server. Điều này đảm bảo ứng dụng hoạt động mượt mà trên nhiều thiết bị với các khả năng phần cứng khác nhau, tối ưu hóa cả hiệu suất và khả năng sử dụng pin.

2.4. Kết chương

Như vậy, Chương 2 đã trình bày một cách hệ thống và chi tiết các cơ sở lý thuyết nền tảng cho việc xây dựng hệ thống điểm danh bằng nhận diện khuôn mặt. Các khái niệm và kỹ thuật cốt lõi trong lĩnh vực nhận diện khuôn mặt, từ phát hiện (MTCNN), tiền xử lý, trích xuất đặc trưng (FaceNet) đến so khớp (Cosine Similarity), đã được phân tích cụ thể. Đồng thời, chương cũng đã giới thiệu các công nghệ và kiến trúc được sử dụng để phát triển phần mềm, bao gồm framework backend Flask với các thành phần như định tuyến, template engine Jinja2, ORM SQLAlchemy, xác thực Flask-Login và quản lý phiên bản Alembic; cùng với framework frontend Flutter cho ứng dụng di động với kiến trúc Widget, cơ chế quản lý trạng thái, giao tiếp API và xử lý ảnh phía client.

Chương 3. PHÂN TÍCH ĐẶC TẢ YÊU CẦU

Trong chương này, chúng ta sẽ tiến hành phân tích chi tiết các yêu cầu chức năng từ góc nhìn của hai tác nhân chính: Giáo viên và Học sinh. Mỗi nhóm chức năng sẽ được mô tả cụ thể, làm rõ các nhiệm vụ và tương tác mà mỗi tác nhân có thể thực hiện với hệ thống. Để trực quan hóa các tương tác này và hiểu rõ hơn về phạm vi của hệ thống, các biểu đồ Use Case tổng quan cho các quy trình chính như quản lý lớp học và điểm danh sẽ được trình bày. Cuối cùng, để làm rõ luồng xử lý chi tiết cho từng chức năng cốt lõi, chương sẽ đi sâu vào các biểu đồ hoạt động, mô tả từng bước thực hiện, các điểm quyết định và luồng dữ liệu cho các ca sử dụng quan trọng như đăng ký, đăng nhập, tải ảnh, điểm danh và xem lịch sử.

3.1. Phân tích yêu cầu

3.1.1. Chức năng cho Actor: Giáo viên

Giáo viên thực hiện đăng ký tài khoản trên giao diện chuyên dụng; tất cả thông tin cá nhân và thông tin xác thực được lưu trữ trong cơ sở dữ liệu bảo mật theo chuẩn OAuth 2.0. Sau khi tài khoản được xác thực, Giáo viên đăng nhập bằng cơ chế phiên (session) được mã hóa, từ đó hệ thống hướng dẫn chuyển tới trang quản lý chính (Dashboard), nơi hiển thị tổng quan các lớp học, bao gồm mã lớp, tên lớp, số lượng sinh viên và trạng thái điểm danh. Tiếp theo, Giáo viên thiết lập lớp học mới bằng cách nhập tên lớp và các tham số liên quan; hệ thống sinh tự động một mã lớp duy nhất theo định dạng alphanumeric 8 ký tự để đảm bảo không trùng lặp. Mọi lớp học do Giáo viên tạo đều được liệt kê theo thứ tự thời gian tạo, kèm tính năng tìm kiếm và phân trang. Khi cần xem thông tin chi tiết của một lớp, Giáo viên truy cập vào trang chi tiết, nơi hệ thống trình bày danh sách sinh viên đã tham gia, thông tin cá nhân cơ bản và thống kê tỷ lệ điểm danh. Trường hợp bổ sung thành viên, Giáo viên thao tác thêm sinh viên thông qua việc nhập ID sinh viên; sau khi xác thực ID tồn tại trong hệ thống, bản ghi quan hệ giữa sinh viên và lớp học được tạo mới trong bảng liên kết. Cuối cùng, Giáo viên truy xuất bản ghi điểm danh của lớp để xem lịch sử điểm danh theo ngày, thời gian và kết quả (có mặt/ vắng mặt), phục vụ cho đánh giá và báo cáo kết quả học tập.

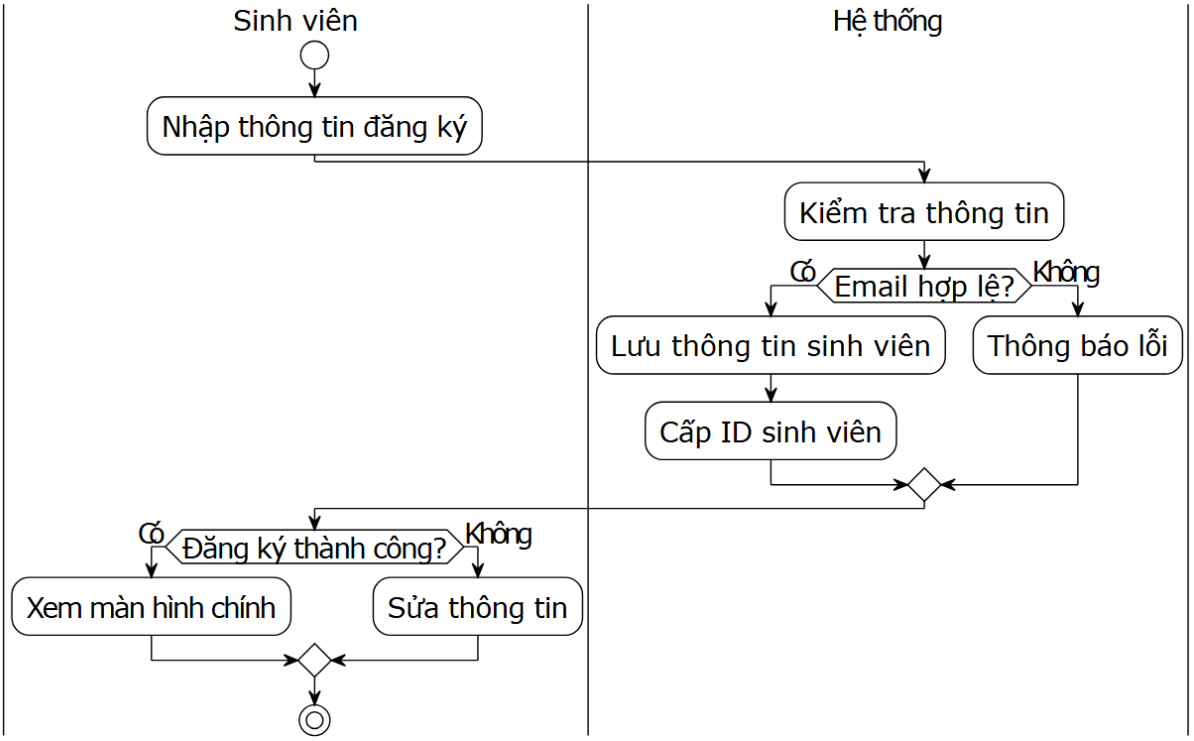
3.1.2. Chức năng cho Actor: Học sinh

Học sinh khởi tạo tài khoản bằng việc cung cấp thông tin cá nhân cơ bản và định danh duy nhất (student ID); hệ thống lưu trữ ảnh chân dung và dữ liệu xác thực dưới

dạng JSON Web Token (JWT) để bảo vệ quyền riêng tư. Sau khi đăng ký thành công, Học sinh đăng nhập vào hệ thống và được điều phối tới trang cá nhân, nơi hiển thị các lớp học đã tham gia. Để tham gia lớp mới, Học sinh nhập mã lớp do Giáo viên cung cấp; hệ thống kiểm tra tính hợp lệ của mã và thực hiện ghi nhận quan hệ tham dự trong cơ sở dữ liệu. Trước mỗi buổi học, Học sinh tải lên ảnh khuôn mặt thông qua giao diện camera tích hợp trong ứng dụng; ảnh được chuyển đến mô-đun nhận diện khuôn mặt (Face Recognition Module) sử dụng FaceNet và MTCNN để so khớp với mẫu ảnh đã lưu. Kết quả xác thực (đúng/ sai) kèm mốc thời gian được hệ thống ghi nhận vào bảng điểm danh. Học sinh có thể truy vấn lịch sử điểm danh cá nhân dưới dạng bảng, trong đó mỗi bản ghi bao gồm ngày, giờ và kết quả điểm danh, hỗ trợ việc đối chiếu thông tin và theo dõi quá trình tham gia học tập.

3.2. Biểu đồ hoạt động các ca sử dụng

Dưới đây là các biểu đồ hoạt động ca sử dụng kèm theo mô tả của các ca sử dụng trong hệ thống điểm danh bằng Facenet



Hình 3.1: Biểu đồ hoạt động sinh viên đăng ký

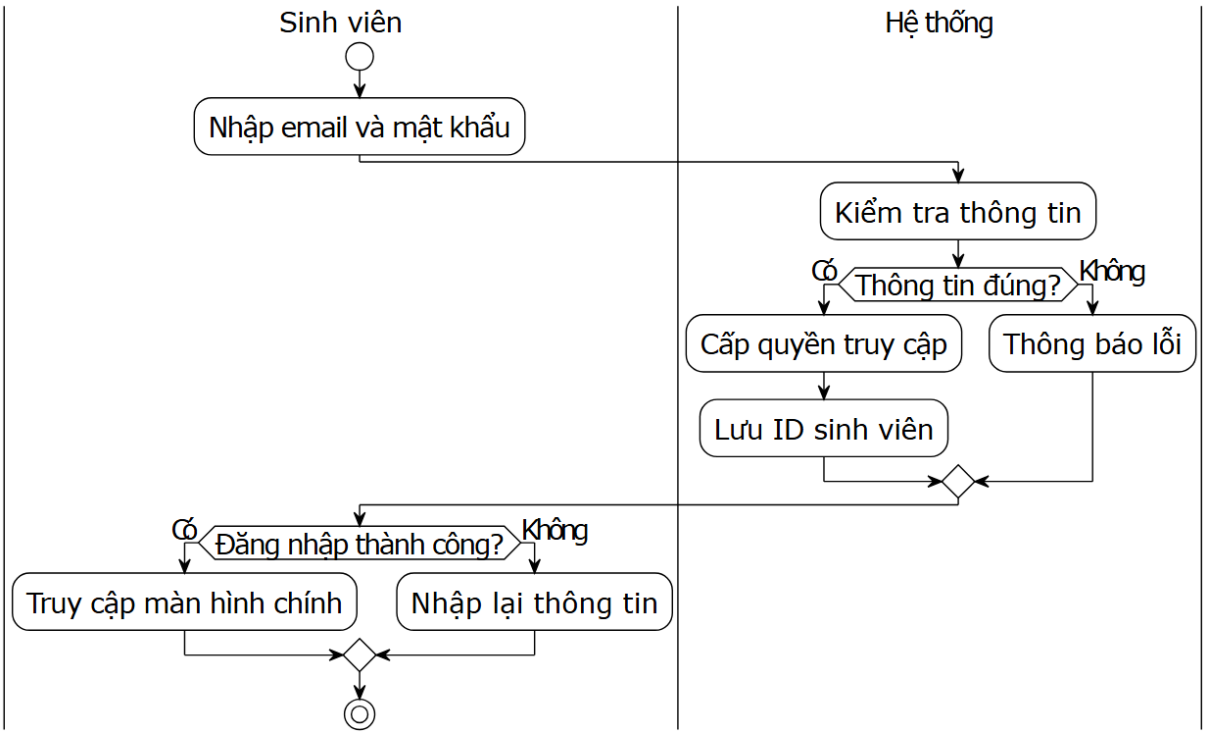
Quy trình đăng ký sinh viên bắt đầu từ phía ứng dụng di động Flutter. Sinh viên mở ứng dụng và đi đến màn hình đăng ký, nơi họ nhập các thông tin cần thiết bao gồm tên người dùng, địa chỉ email, mật khẩu và mã lớp học. Sau khi hoàn thành việc nhập

liệu, sinh viên nhấn nút đăng ký, khởi tạo yêu cầu HTTP được gửi đến API của hệ thống điểm danh.

Khi nhận được yêu cầu, hệ thống backend Flask bắt đầu xử lý thông qua một chuỗi các bước kiểm tra và xác thực. Đầu tiên, hệ thống kiểm tra định dạng email để đảm bảo tính hợp lệ. Tiếp theo, hệ thống tìm kiếm trong cơ sở dữ liệu để xác định xem địa chỉ email này đã được sử dụng trước đó hay chưa. Nếu email đã tồn tại, hệ thống sẽ phản hồi với thông báo lỗi "Email đã được sử dụng" và yêu cầu sinh viên sử dụng địa chỉ email khác hoặc đăng nhập nếu đã có tài khoản.

Trong trường hợp email hợp lệ và chưa được sử dụng, hệ thống tiếp tục tiến hành các bước xử lý tiếp theo. Mật khẩu được băm an toàn bằng thuật toán mã hóa một chiều trước khi lưu trữ, đảm bảo tính bảo mật của dữ liệu. Thông tin sinh viên mới sau đó được lưu vào cơ sở dữ liệu.

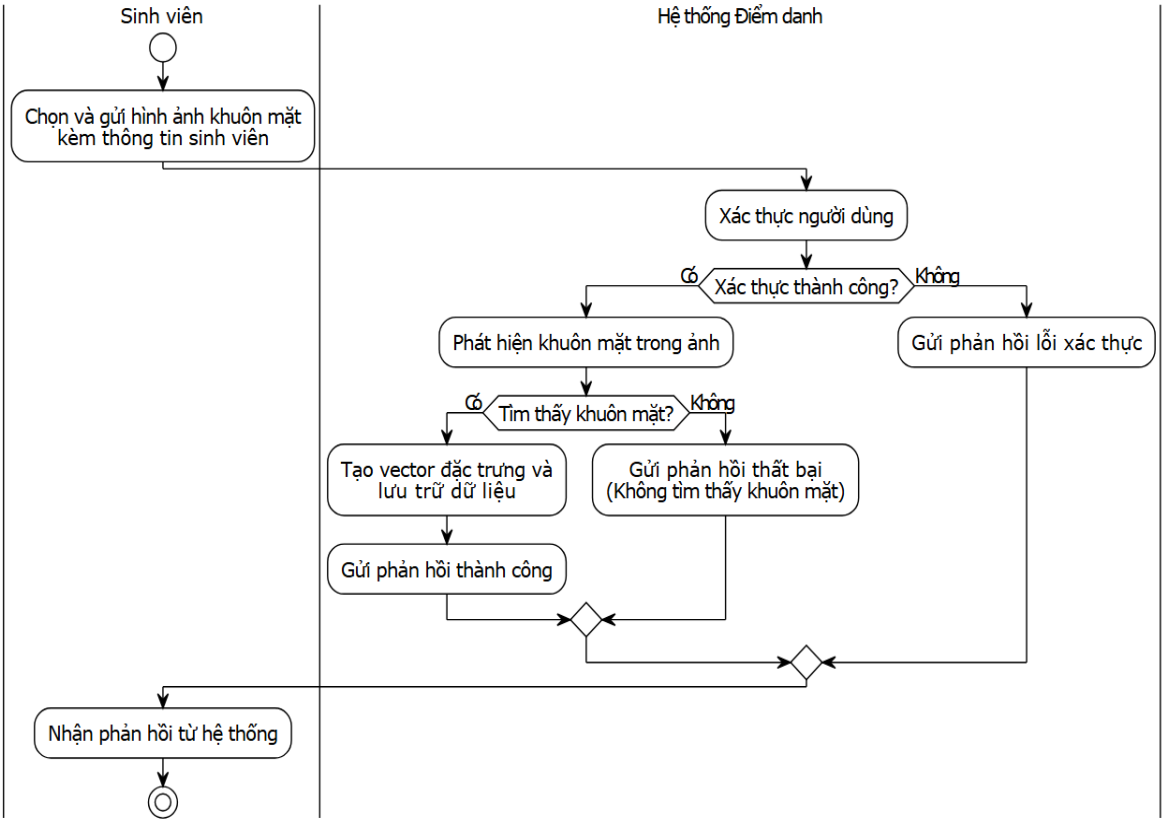
Sau khi lưu thành công, hệ thống tạo một ID sinh viên duy nhất và lưu trữ vào cơ sở dữ liệu. Quá trình hoàn tất với việc hệ thống gửi phản hồi thành công về cho ứng dụng di động, kèm theo ID sinh viên vừa tạo. Khi nhận được phản hồi thành công này, ứng dụng di động lưu trữ ID sinh viên vào bộ nhớ cục bộ thông qua SharedPreferences để sử dụng cho các phiên làm việc sau, và hiển thị thông báo đăng ký thành công cho sinh viên.



Hình 3.2: Biểu đồ hoạt động sinh viên đăng nhập

Quá trình đăng nhập sinh viên trong hệ thống điểm danh bắt đầu khi người dùng nhập email và mật khẩu trên ứng dụng di động, sau đó gửi yêu cầu xác thực đến backend Flask thông qua API. Hệ thống backend nhận yêu cầu và thực hiện hai bước kiểm tra quan trọng: đầu tiên tìm kiếm sinh viên dựa trên email, sau đó so sánh mật khẩu được cung cấp với mật khẩu đã băm trong cơ sở dữ liệu. Quá trình xác thực này đảm bảo chỉ những sinh viên đã đăng ký mới có thể truy cập vào hệ thống.

Khi xác thực thành công, backend tạo phiên làm việc cho sinh viên và gửi về thông tin chi tiết bao gồm ID sinh viên, tên, thông tin lớp học và trạng thái mã hóa khuôn mặt. Ứng dụng di động sau đó lưu trữ ID sinh viên vào bộ nhớ cục bộ của thiết bị, thiết lập phiên đăng nhập và sử dụng ID này để xác thực các yêu cầu API tiếp theo. Cơ chế này cho phép sinh viên duy trì trạng thái đăng nhập ngay cả khi đóng và mở lại ứng dụng.



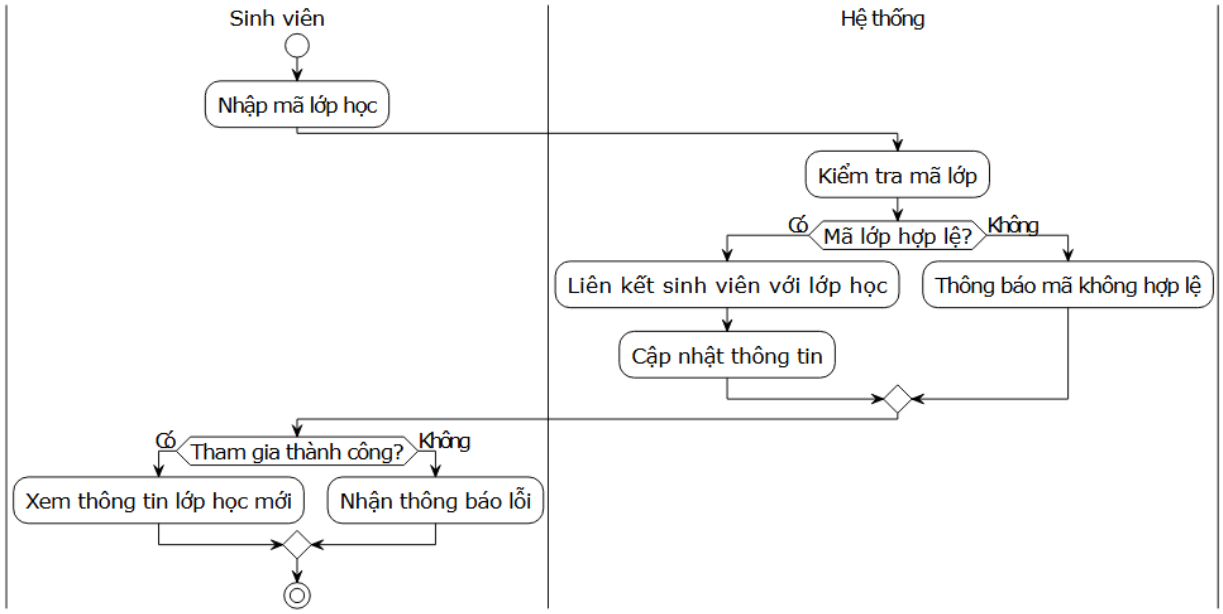
Hình 3.3: Biểu đồ hoạt động sinh viên gửi hình ảnh khuôn mặt

Quá trình bắt đầu khi sinh viên đã đăng nhập vào ứng dụng Flutter và truy cập màn hình tải lên khuôn mặt. Tại đây, sinh viên chọn một hoặc nhiều ảnh khuôn mặt từ thư viện thiết bị hoặc sử dụng camera để chụp ảnh trực tiếp. Ứng dụng hiển thị các ảnh đã chọn để sinh viên xác nhận trước khi gửi đến hệ thống backend.

Khi sinh viên xác nhận và gửi ảnh, ứng dụng đóng gói ID sinh viên cùng với các

ảnh đã chọn thành yêu cầu HTTP đa phần (multipart) và gửi đến server qua endpoint đăng ký. Server xác thực thông tin sinh viên và lưu trữ các ảnh vào thư mục cấu trúc theo lớp học và ID sinh viên. Sau đó, hệ thống phân tích từng ảnh để phát hiện khuôn mặt và từ chối những ảnh không có khuôn mặt rõ ràng hoặc có chất lượng thấp.

Với các ảnh hợp lệ, hệ thống tạo ra vector đặc trưng cho mỗi khuôn mặt và tính toán một vector trung bình đại diện cho sinh viên đó. Vector này được lưu trữ trong cơ sở dữ liệu embeddings của lớp học tương ứng và trạng thái hoàn thành mã hóa khuôn mặt của sinh viên được cập nhật. Sau khi hoàn tất, hệ thống thông báo kết quả thành công về ứng dụng, và sinh viên đã sẵn sàng để sử dụng tính năng điểm danh bằng nhận diện khuôn mặt trong các buổi học tiếp theo.

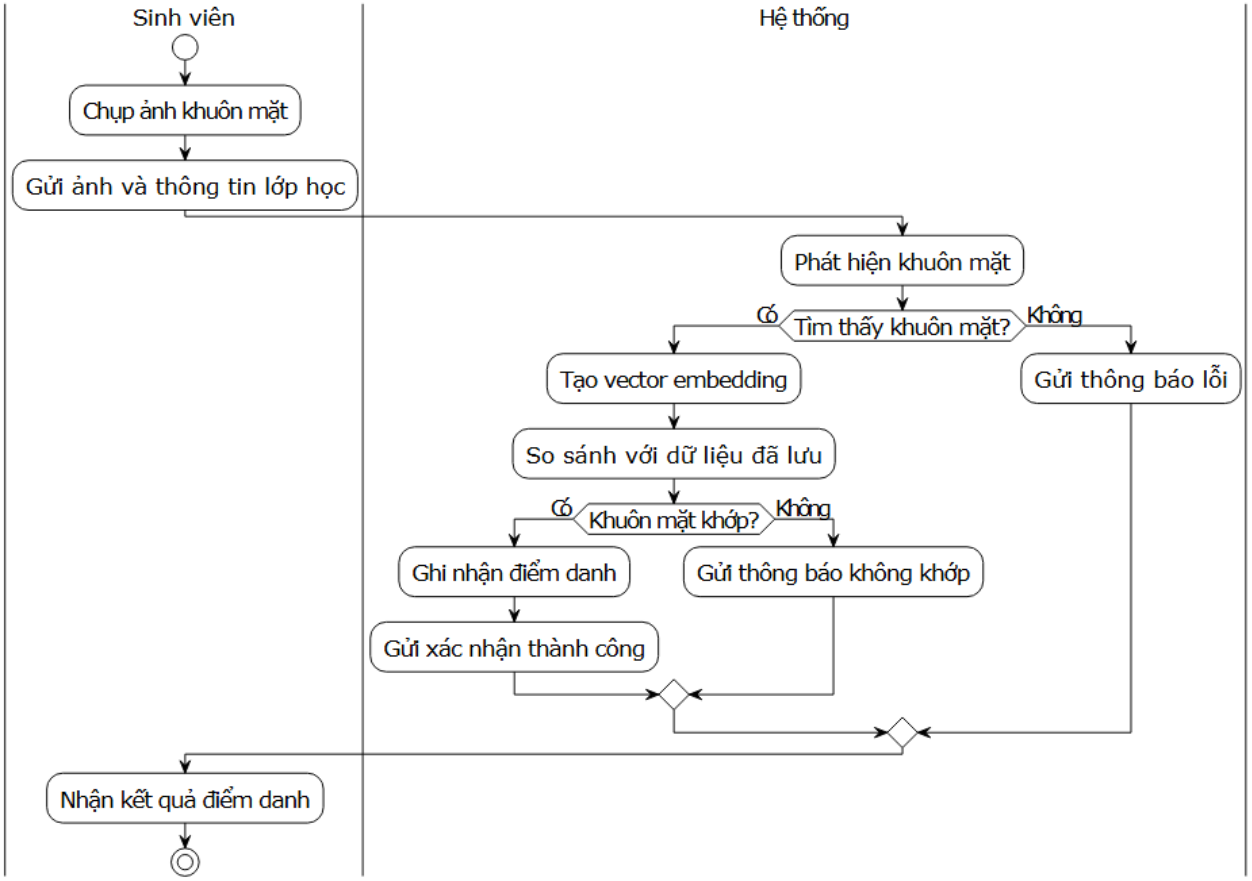


Hình 3.4: Biểu đồ hoạt động sinh viên tham gia lớp học

Quá trình tham gia lớp học bắt đầu khi sinh viên đã đăng nhập vào ứng dụng di động và chọn chức năng "Tham gia lớp học". Tại đây, sinh viên nhập mã lớp học đã được giáo viên cung cấp trước đó, sau đó nhấn nút gửi để tạo yêu cầu tham gia. Ứng dụng tạo một yêu cầu HTTP POST đến endpoint, kèm theo ID sinh viên và mã lớp học cần tham gia.

Khi nhận được yêu cầu, hệ thống backend trước tiên xác thực sinh viên thông qua ID đã gửi kèm. Tiếp theo, hệ thống kiểm tra tính hợp lệ của mã lớp học bằng cách tìm kiếm trong cơ sở dữ liệu. Nếu không tìm thấy lớp học với mã được cung cấp, hệ thống sẽ phản hồi với thông báo lỗi "Mã lớp không hợp lệ" và quá trình kết thúc, yêu cầu sinh viên kiểm tra lại mã lớp.

Khi xác nhận mã lớp hợp lệ, hệ thống tiến hành liên kết sinh viên với lớp học thông qua việc cập nhật trường class_id trong bản ghi của sinh viên. Đặc biệt, nếu sinh viên đã hoàn thành việc đăng ký khuôn mặt trước đó, hệ thống sẽ tự động chuyển dữ liệu embedding khuôn mặt sang lớp học mới để đảm bảo tính liên tục của chức năng nhận diện. Sau khi cập nhật thành công, hệ thống gửi phản hồi về ứng dụng với thông tin chi tiết về lớp học đã tham gia, cho phép sinh viên bắt đầu sử dụng các tính năng điểm danh trong lớp học mới ngay lập tức.



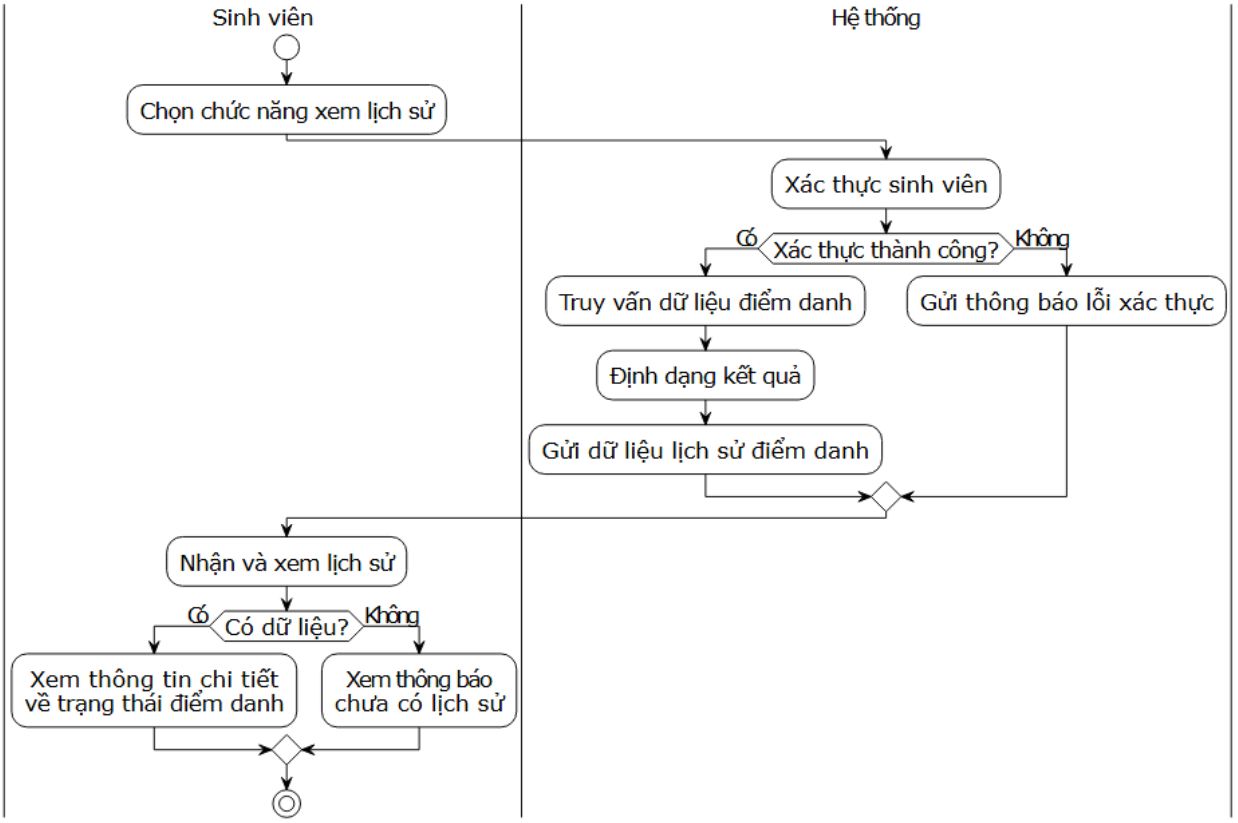
Hình 3.5: Biểu đồ hoạt động sinh viên thực hiện điểm danh

Quy trình điểm danh của sinh viên bắt đầu khi họ đã đăng nhập vào ứng dụng di động và chọn chức năng "Điểm danh". Tại đây, sinh viên được yêu cầu chụp ảnh selfie hoặc chọn ảnh có sẵn từ thư viện, sau đó nhấn nút gửi để tiến hành điểm danh. Ứng dụng đóng gói ID sinh viên, ID lớp học và ảnh khuôn mặt trong một yêu cầu đến endpoint của hệ thống.

Khi nhận được yêu cầu, hệ thống backend trước tiên xác thực danh tính sinh viên và kiểm tra xem họ có thuộc về lớp học được chỉ định không. Sau đó, hệ thống sử dụng MTCNN để phát hiện khuôn mặt trong ảnh được gửi lên. Nếu không tìm thấy khuôn mặt hoặc độ tin cậy phát hiện thấp, hệ thống sẽ phản hồi với thông báo lỗi và

yêu cầu sinh viên thử lại với ảnh chất lượng tốt hơn. Nếu phát hiện thành công, khuôn mặt được tiền xử lý và chuyển đổi thành vector embedding 512 chiều bằng mô hình FaceNet.

Sau khi có được embedding, hệ thống gọi hàm để so sánh vector này với embedding đã lưu trữ của sinh viên, sử dụng độ tương đồng cosine với ngưỡng 0.6. Nếu khuôn mặt khớp, hệ thống tạo hoặc cập nhật bản ghi điểm danh trong cơ sở dữ liệu với trạng thái "có mặt" cùng thời gian hiện tại, sau đó gửi phản hồi thành công về ứng dụng. Nếu xác thực khuôn mặt thất bại, sinh viên sẽ nhận được thông báo lỗi và được yêu cầu thử lại. Quá trình điểm danh này đảm bảo chỉ sinh viên thực sự có mặt mới có thể điểm danh thành công, ngăn chặn các trường hợp điểm danh hộ.



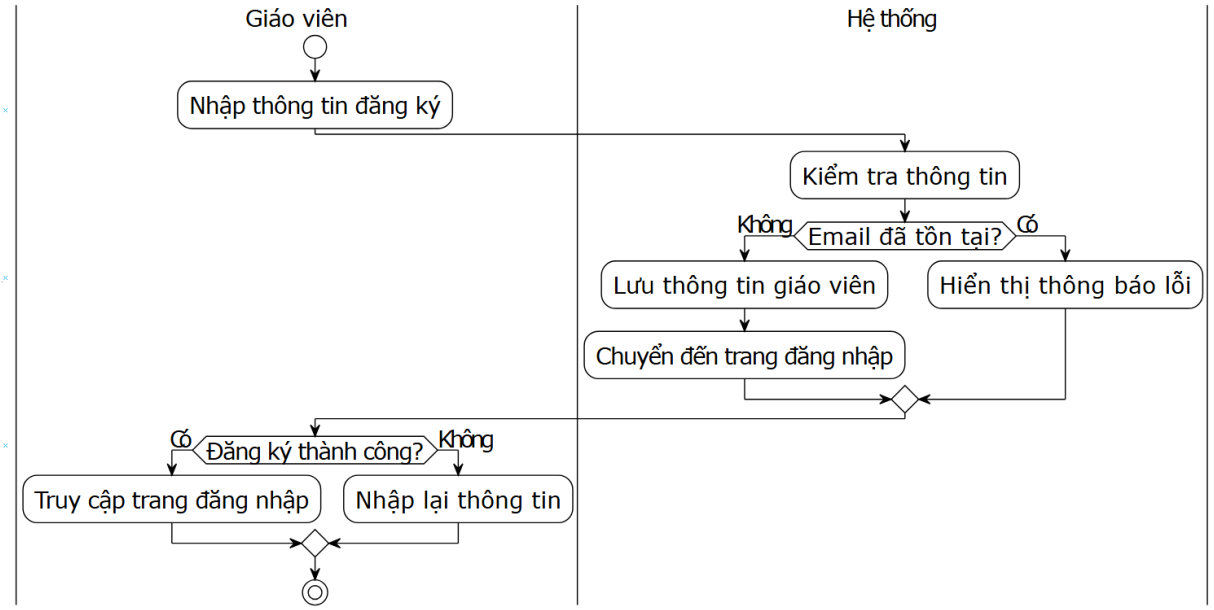
Hình 3.6: Biểu đồ hoạt động sinh viên xem lịch sử điểm danh

Quy trình xem lịch sử điểm danh bắt đầu khi sinh viên đăng nhập vào ứng dụng di động và chọn chức năng "Xem lịch sử điểm danh" từ menu chính. Thao tác này kích hoạt một yêu cầu đến endpoint, nơi hệ thống sẽ xác thực danh tính sinh viên thông qua ID được lưu trữ trong bộ nhớ cục bộ của thiết bị. Nếu xác thực thất bại, sinh viên sẽ nhận được thông báo lỗi và được yêu cầu đăng nhập lại để tiếp tục.

Sau khi xác thực thành công, hệ thống backend truy vấn cơ sở dữ liệu để lấy tất cả bản ghi điểm danh liên quan đến sinh viên đó, sắp xếp theo thứ tự thời gian giảm

dần để hiển thị các mục mới nhất trước. Dữ liệu được truy xuất bao gồm ngày điểm danh, trạng thái (có mặt/vắng mặt), thời gian chính xác và tên lớp học tương ứng. Hệ thống đóng gói thông tin này thành định dạng JSON và gửi phản hồi về ứng dụng di động.

Khi nhận được phản hồi từ server, ứng dụng xử lý dữ liệu JSON và hiển thị lịch sử điểm danh dưới dạng danh sách có cấu trúc. Với mỗi bản ghi điểm danh, sinh viên có thể thấy ngày tháng, trạng thái điểm danh và tên lớp học. Nếu không có dữ liệu điểm danh nào được tìm thấy, ứng dụng sẽ hiển thị thông báo "Chưa có lịch sử điểm danh". Giao diện trực quan này giúp sinh viên dễ dàng theo dõi tình trạng tham gia học tập của mình trong suốt khóa học.

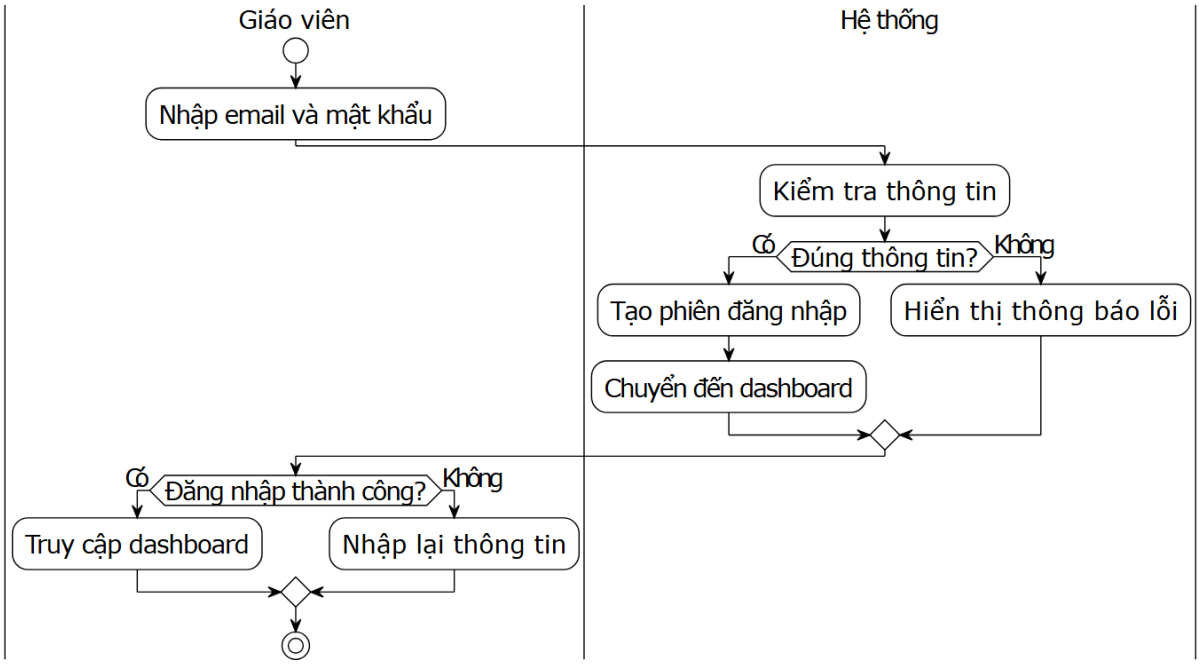


Hình 3.7: Biểu đồ hoạt động giáo viên đăng ký tài khoản

Quy trình đăng ký tài khoản giáo viên bắt đầu khi người dùng truy cập trang web của hệ thống điểm danh và chọn chức năng "Đăng ký". Tại đây, giáo viên điền các thông tin cần thiết vào form đăng ký, bao gồm tên đăng nhập), địa chỉ email và mật khẩu. Sau khi hoàn tất nhập liệu, giáo viên nhấn nút đăng ký để gửi thông tin đến server Flask thông qua phương thức HTTP POST.

Khi nhận được yêu cầu, hệ thống backend tiến hành kiểm tra tính hợp lệ của thông tin đăng ký. Đầu tiên, Flask-WTF kiểm tra xem các trường dữ liệu đã được điền đầy đủ và đúng định dạng hay chưa. Tiếp theo, hệ thống truy vấn cơ sở dữ liệu để đảm bảo tên đăng nhập và địa chỉ email chưa được sử dụng bởi giáo viên khác. Nếu phát hiện bất kỳ trùng lặp nào, hệ thống sẽ hiển thị thông báo lỗi tương ứng và yêu cầu giáo viên cung cấp thông tin khác.

Trong trường hợp thông tin đăng ký hợp lệ và độc nhất, hệ thống tạo một đối tượng Teacher mới, mã hóa mật khẩu sau đó lưu thông tin vào cơ sở dữ liệu. Khi quá trình lưu trữ hoàn tất, hệ thống hiển thị thông báo đăng ký thành công và chuyển hướng giáo viên đến trang đăng nhập. Tại đây, giáo viên có thể sử dụng thông tin vừa đăng ký để đăng nhập và bắt đầu sử dụng các chức năng quản lý lớp học và điểm danh của hệ thống.



Hình 3.8: Biểu đồ hoạt động giáo viên đăng nhập tài khoản

Quy trình đăng nhập tài khoản giáo viên bắt đầu khi giáo viên truy cập trang web của hệ thống điểm danh và chọn chức năng "Đăng nhập". Tại đây, giáo viên nhập địa chỉ email và mật khẩu đã đăng ký trước đó vào form đăng nhập, sau đó nhấn nút đăng nhập để gửi thông tin xác thực đến server Flask thông qua phương thức HTTP POST. Form này được xử lý bởi Flask-WTF để đảm bảo các trường được điền đầy đủ và phù hợp với định dạng yêu cầu.

Khi nhận được yêu cầu, hệ thống backend xử lý thông tin đăng nhập thông qua các bước xác thực tuần tự. Đầu tiên, hệ thống truy vấn cơ sở dữ liệu để tìm kiếm giáo viên với địa chỉ email được cung cấp. Nếu không tìm thấy, hoặc nếu tìm thấy nhưng mật khẩu không khớp khi kiểm tra, hệ thống sẽ hiển thị thông báo lỗi "Email hoặc mật khẩu không hợp lệ" và yêu cầu giáo viên thử lại.

Trong trường hợp xác thực thành công, hệ thống tạo một phiên đăng nhập mới bằng cách sử dụng Flask-Login, lưu trữ thông tin giáo viên đang đăng nhập và thiết lập cookie phiên làm việc. Sau đó, hệ thống chuyển hướng giáo viên đến trang dashboard,

nơi họ có thể thấy danh sách các lớp học đã tạo và truy cập các chức năng quản lý. Phiên đăng nhập này sẽ được duy trì cho đến khi giáo viên chủ động đăng xuất hoặc phiên hết hạn, giúp đảm bảo trải nghiệm sử dụng liền mạch và an toàn.

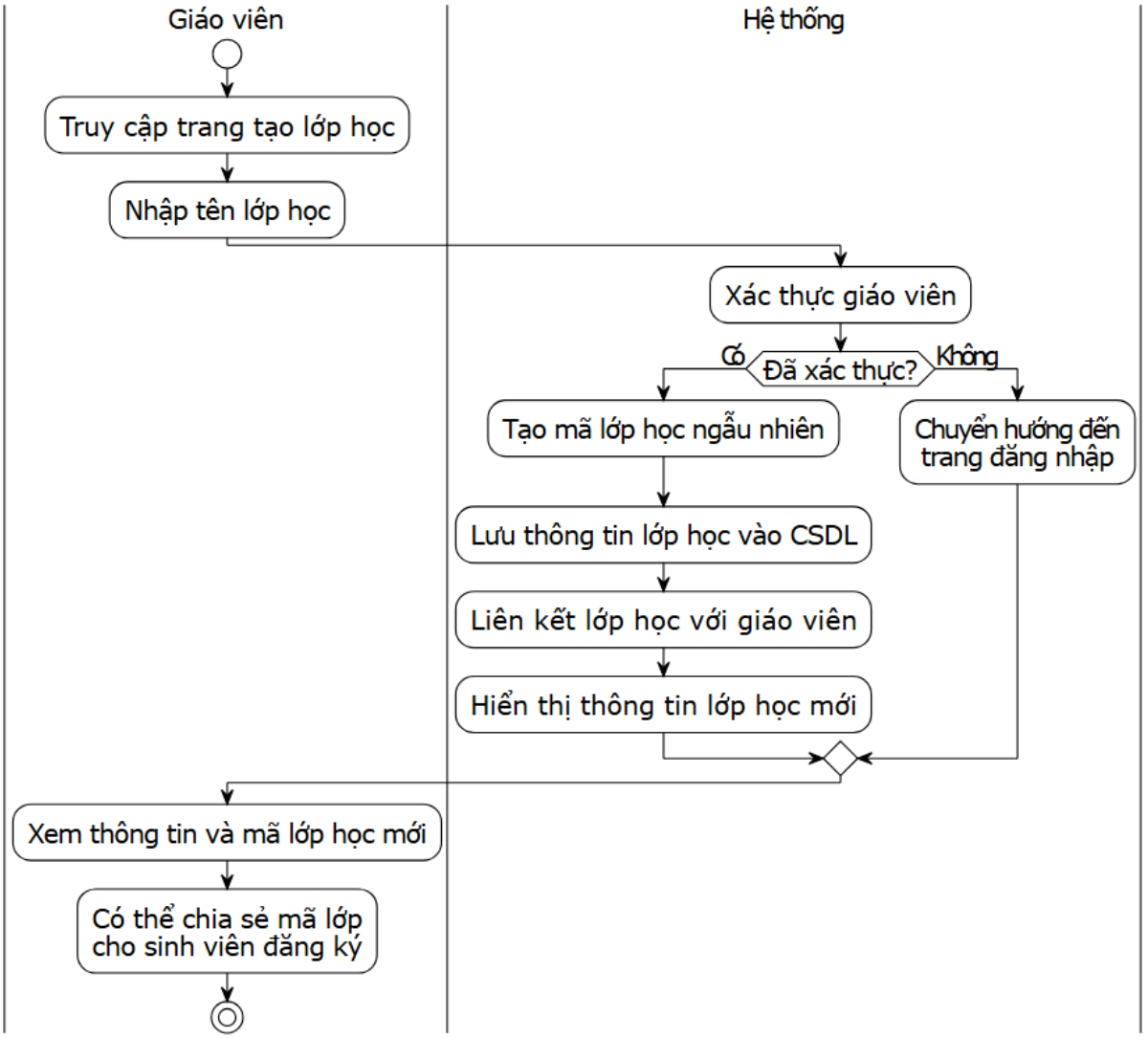
Hình 3.9: Biểu đồ hoạt động giáo viên đăng xuất tài khoản

Quá trình đăng xuất tài khoản giáo viên bắt đầu khi giáo viên đã đăng nhập vào hệ thống và nhấn vào liên kết "Đăng xuất" trên thanh điều hướng của trang web. Yêu cầu này được xử lý bởi route đăng xuất đã được đăng ký, đảm bảo chỉ những người dùng đã đăng nhập mới có thể truy cập vào chức năng này.

Khi server nhận được yêu cầu đăng xuất hợp lệ, nó gọi một phương thức từ thư viện Flask-Login. Phương thức này thực hiện một loạt các thao tác nền để xóa thông tin phiên làm việc của người dùng: hủy bỏ cookie phiên, xóa thông tin người dùng hiện tại khỏi bộ nhớ của server, và vô hiệu hóa tất cả các token xác thực đang được sử dụng. Quá trình này đảm bảo rằng các yêu cầu tiếp theo từ trình duyệt không còn được xem là đã xác thực.

Sau khi hoàn tất việc xóa phiên làm việc, hệ thống chuyển hướng giáo viên về trang chủ, nơi họ sẽ thấy tùy chọn đăng nhập lại. Toàn bộ quy trình đăng xuất này diễn ra một cách nhanh chóng và liền mạch, yêu cầu tối thiểu sự tương tác từ người dùng nhưng vẫn đảm bảo rằng thông tin đăng nhập và phiên làm việc được xóa bỏ một cách

an toàn, bảo vệ tài khoản giáo viên khỏi việc truy cập trái phép từ cùng một thiết bị hoặc trình duyệt.

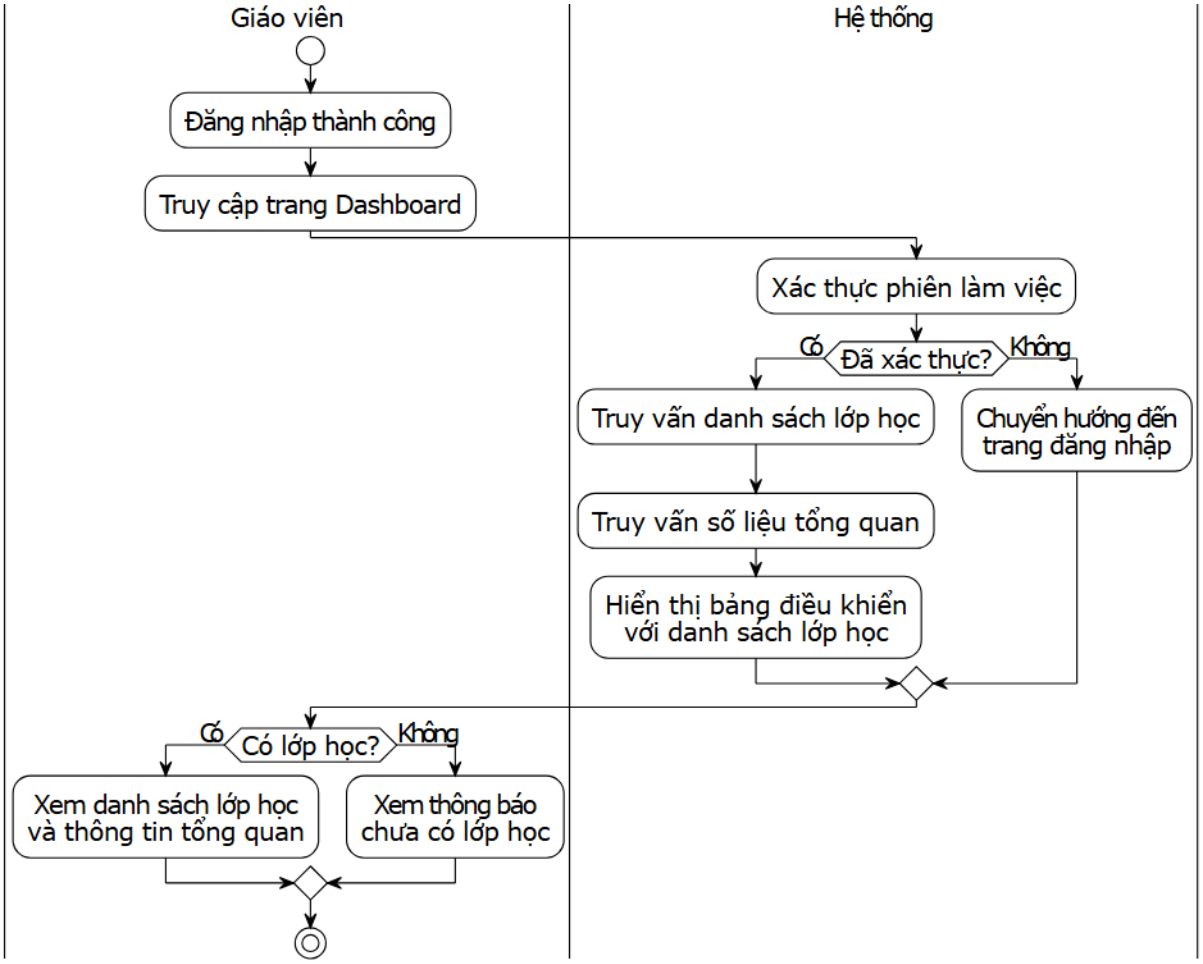


Hình 3.10: Biểu đồ hoạt động giáo viên tạo lớp học mới

Quy trình tạo lớp học mới bắt đầu khi giáo viên đã đăng nhập vào hệ thống web và truy cập chức năng "Tạo lớp học mới" từ dashboard hoặc menu quản lý. Tại đây, giáo viên được hiển thị một form đơn giản yêu cầu nhập tên lớp học. Sau khi điền thông tin và nhấn nút "Tạo lớp", hệ thống gửi yêu cầu HTTP POST đến endpoint kèm theo tên lớp học và ID giáo viên đã được xác thực.

Khi nhận được yêu cầu, hệ thống backend xác thực người dùng, đảm bảo chỉ giáo viên đã đăng nhập mới có thể tạo lớp học mới. Sau đó, hệ thống tạo một đối tượng mới với tên lớp được cung cấp và tự động sinh mã lớp độc nhất dài 8 ký tự. Mã lớp này đóng vai trò quan trọng để sinh viên có thể tham gia vào lớp học từ ứng dụng di động. Lớp học mới được liên kết với giáo viên hiện tại.

Sau khi lưu thông tin lớp học vào cơ sở dữ liệu, hệ thống chuyển hướng giáo viên đến trang chi tiết lớp học mới tạo, nơi hiển thị thông tin như tên lớp, mã lớp và các tùy chọn quản lý. Mã lớp được hiển thị nổi bật để giáo viên có thể dễ dàng chia sẻ với sinh viên thông qua email, tin nhắn hoặc thông báo trực tiếp trong lớp học. Từ đây, giáo viên có thể tiếp tục thêm sinh viên vào lớp hoặc tiến hành các hoạt động điểm danh cho lớp học vừa tạo.



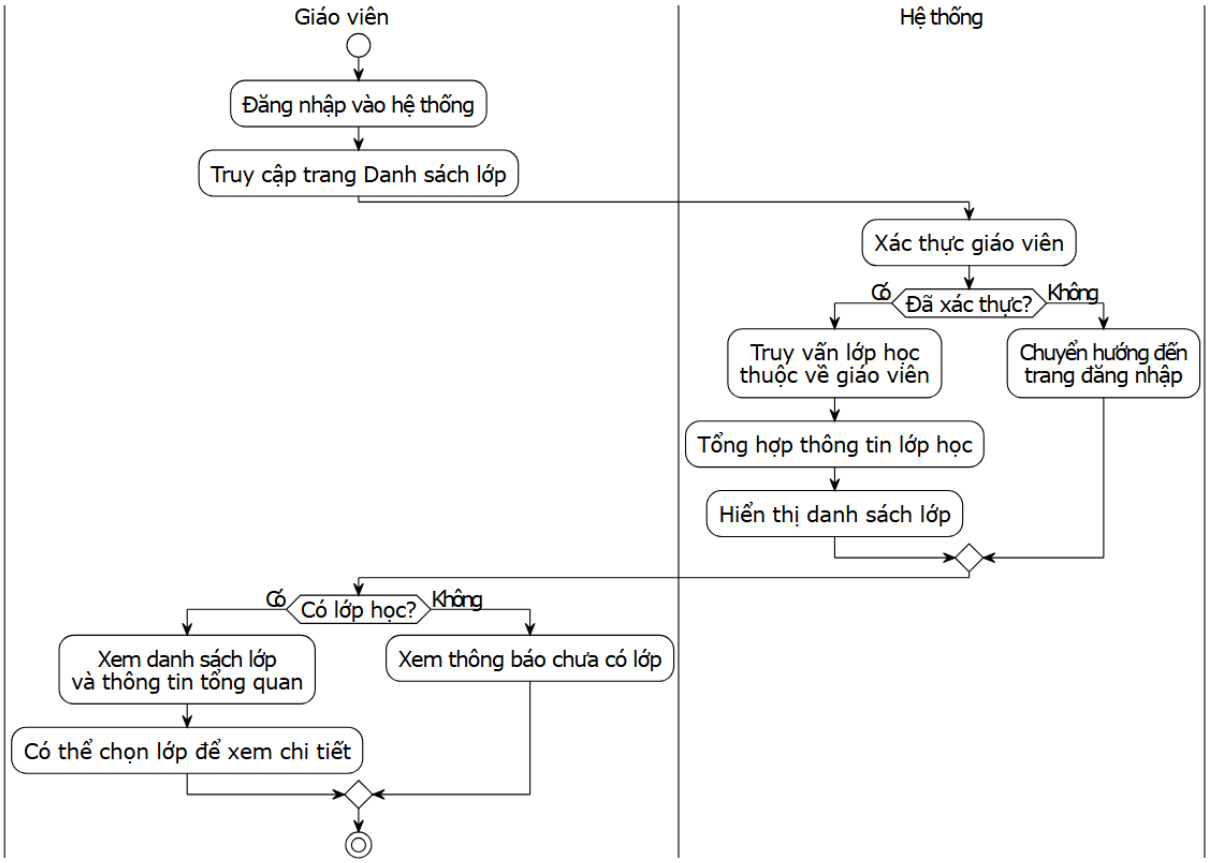
Hình 3.11: Biểu đồ hoạt động giáo viên xem bảng điều khiển

Quy trình xem bảng điều khiển của giáo viên bắt đầu khi giáo viên đăng nhập thành công vào hệ thống web và được chuyển hướng tự động đến trang dashboard, hoặc khi họ chọn tùy chọn "Dashboard" từ menu điều hướng. Tại thời điểm này, hệ thống gửi yêu cầu HTTP GET đến điểm cuối. Trước khi hiển thị nội dung, hệ thống xác thực trạng thái đăng nhập của giáo viên, đảm bảo chỉ những người dùng đã xác thực mới có thể truy cập bảng điều khiển.

Sau khi xác nhận giáo viên đã đăng nhập, backend truy vấn cơ sở dữ liệu để lấy danh sách các lớp học thuộc về giáo viên hiện tại thông qua mối quan hệ được định

nghĩa trước. Với mỗi lớp học, hệ thống tính toán các thông số tổng quan như tổng số sinh viên, số lượng điểm danh gần đây, và các số liệu thống kê khác để cung cấp một cái nhìn tổng quan về hoạt động của từng lớp. Các thông tin này được đóng gói và truyền vào template dashboard.html để hiển thị.

Khi bảng điều khiển được tạo, giáo viên nhìn thấy bảng điều khiển với các thẻ hiển thị thông tin tổng quan về mỗi lớp học, bao gồm tên lớp, mã lớp, số lượng sinh viên, và tỷ lệ điểm danh. Từ đây, giáo viên có thể nhanh chóng đánh giá tình hình các lớp học của mình và lựa chọn các hành động tiếp theo như tạo lớp học mới, xem chi tiết một lớp cụ thể, hoặc tiến hành điểm danh. Giao diện trực quan này giúp giáo viên quản lý hiệu quả các hoạt động giảng dạy và theo dõi sự tham gia của sinh viên trong từng lớp học.



Hình 3.12: Biểu đồ hoạt động giáo viên xem bảng điều khiển

Luồng bắt đầu khi giáo viên đăng nhập thành công vào hệ thống web và chọn chức năng "Danh sách lớp học". Thao tác này kích hoạt một yêu cầu HTTP GET đến đường dẫn "/classes", được xử lý bởi hàm list_classes() trong tệp classes.py của hệ thống Flask backend.

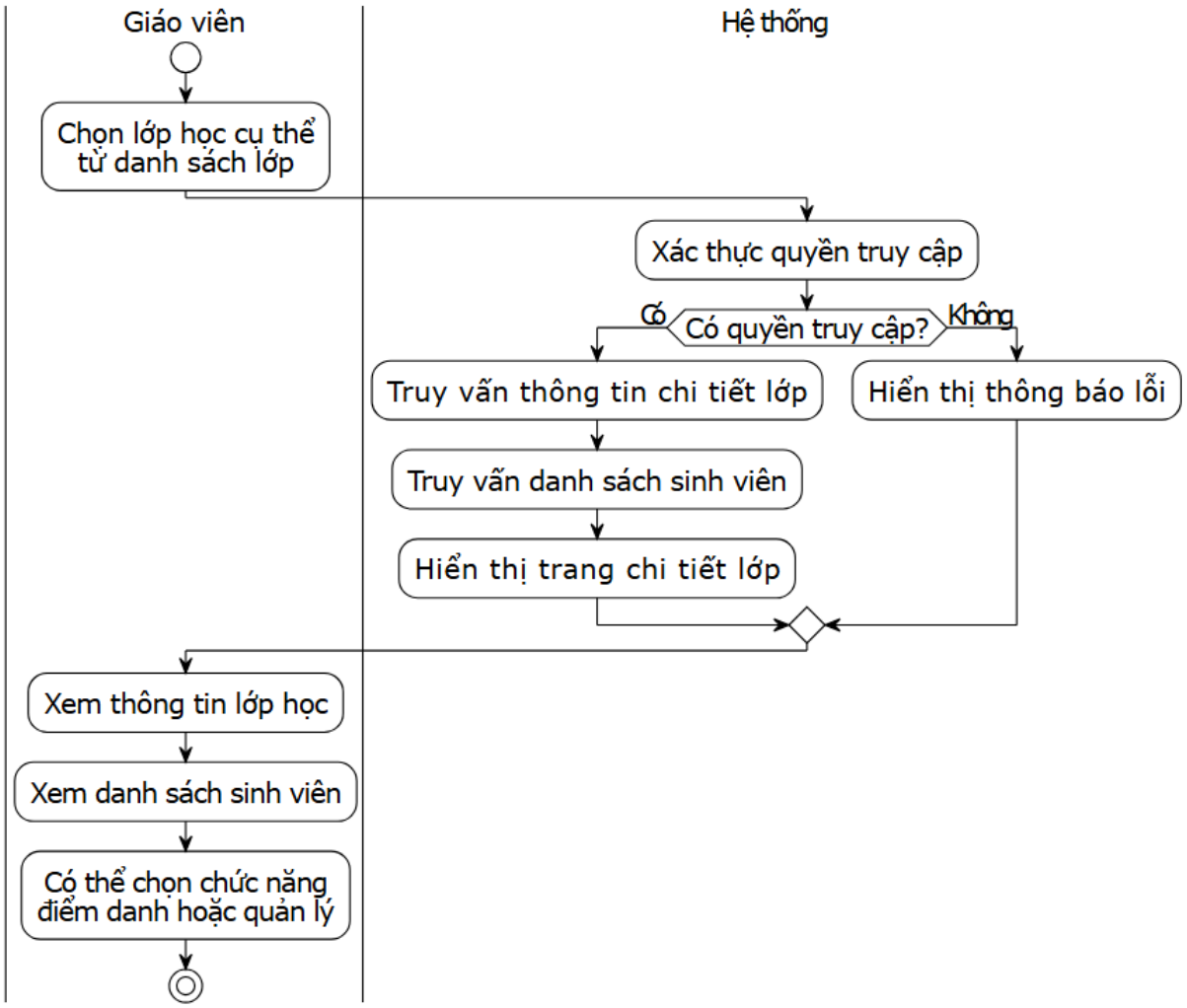
Khi yêu cầu được gửi đến server, bước đầu tiên trong xử lý là kiểm tra xác thực

thông qua decorator @login_required. Điểm quyết định này là thiết yếu trong hệ thống, vì nó đảm bảo rằng chỉ giáo viên đã đăng nhập mới có thể truy cập dữ liệu lớp học, phản ánh cơ chế bảo mật của ứng dụng. Nếu giáo viên chưa đăng nhập hoặc phiên đăng nhập đã hết hạn, hệ thống sẽ chuyển hướng đến trang đăng nhập, yêu cầu xác thực lại trước khi tiếp tục.

Sau khi xác thực thành công, hệ thống truy vấn cơ sở dữ liệu attendance.db để lấy thông tin lớp học thuộc về giáo viên hiện tại thông qua current_user.classes, một mối quan hệ được định nghĩa trong app/models.py. Đặc biệt quan trọng là biểu đồ thể hiện điểm quyết định kiểm tra xem giáo viên có lớp học nào hay không. Nếu không có lớp nào, hệ thống sẽ hiển thị thông báo "Chưa có lớp học" và đề xuất tạo lớp mới. Ngược lại, nếu tìm thấy lớp học, hệ thống sẽ tổng hợp thông tin chi tiết về mỗi lớp, bao gồm số lượng sinh viên và thống kê điểm danh.

Dữ liệu được tổng hợp sau đó được truyền vào template classes/list.html, hiển thị cho giáo viên danh sách lớp học dưới dạng các thẻ hoặc bảng có cấu trúc. Mỗi lớp học hiển thị thông tin cơ bản như tên lớp, mã lớp, số lượng sinh viên đã tham gia và các thống kê điểm danh. Thiết kế giao diện này được tối ưu hóa để cung cấp cái nhìn tổng quan nhanh chóng về tất cả các lớp học của giáo viên.

Điểm cuối của biểu đồ thể hiện khả năng tương tác sau khi danh sách lớp được hiển thị. Giáo viên có thể thực hiện nhiều hành động khác nhau: chọn lớp học cụ thể để xem chi tiết (dẫn đến luồng hoạt động riêng biệt), tạo lớp học mới thông qua nút "Tạo lớp mới", hoặc quay lại dashboard chính. Tính năng này phản ánh thiết kế hướng người dùng của hệ thống, cho phép giáo viên dễ dàng chuyển đổi giữa các chức năng quản lý khác nhau, tối ưu hóa quy trình làm việc trong môi trường giáo dục.

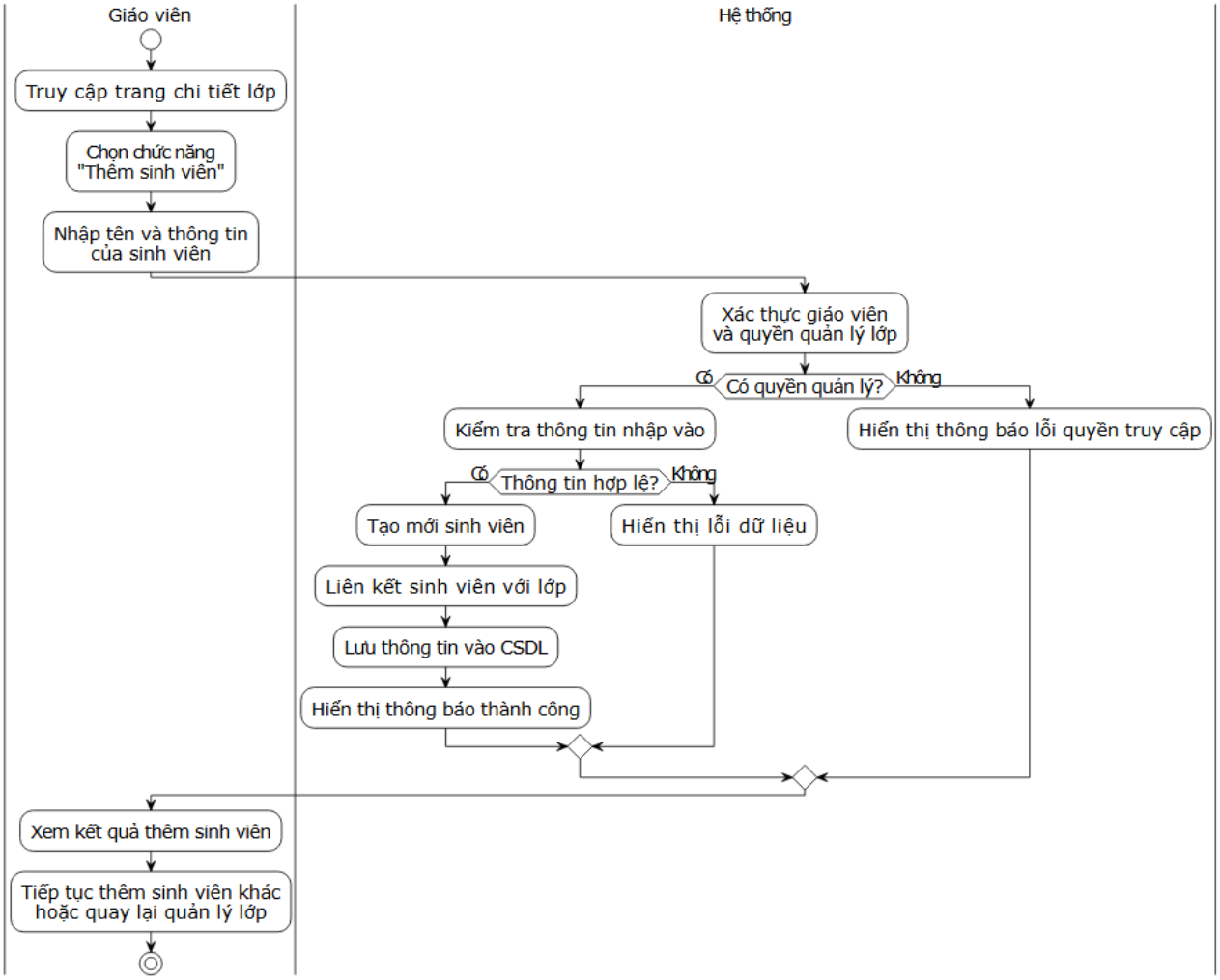


Hình 3.13: Biểu đồ hoạt động giáo viên xem chi tiết lớp

Quy trình xem danh sách lớp học bắt đầu khi giáo viên đã đăng nhập vào hệ thống web và truy cập chức năng "Danh sách lớp học" từ thanh điều hướng hoặc bảng điều khiển chính. Thao tác này tạo một yêu cầu tới máy chủ, được xử lý bởi thành phần điều hướng trong mã nguồn. Hệ thống kiểm tra tình trạng xác thực của giáo viên để đảm bảo rằng chỉ người dùng đã đăng nhập mới có thể xem được thông tin này, nhằm bảo vệ dữ liệu lớp học khỏi những người không có quyền truy cập.

Sau khi xác minh thành công, hệ thống truy vấn cơ sở dữ liệu để lấy danh sách các lớp học thuộc về giáo viên đang đăng nhập thông qua mối quan hệ đã được thiết lập trong mô hình dữ liệu. Đối với mỗi lớp học, hệ thống thu thập thêm các thông tin bổ sung như số lượng học sinh, ngày tạo lớp, và các thông số thống kê về tình hình điểm danh gần đây. Toàn bộ dữ liệu này được sắp xếp và chuyển vào biểu mẫu hiển thị để tạo nên giao diện người dùng.

Giao diện danh sách lớp học hiển thị cho giáo viên dưới dạng bảng hoặc danh sách các thẻ thông tin, mỗi thẻ đại diện cho một lớp học với các chi tiết như tên lớp, mã lớp, số lượng học sinh, và thời gian tạo. Từ đây, giáo viên có thể thực hiện nhiều thao tác khác nhau như xem chi tiết một lớp cụ thể bằng cách nhấn vào tên lớp, tạo lớp học mới thông qua nút chức năng, hoặc truy cập trực tiếp vào phần điểm danh cho một lớp. Thiết kế trực quan này giúp giáo viên dễ dàng quản lý nhiều lớp học cùng lúc và nhanh chóng chuyển đổi giữa các nhiệm vụ quản lý khác nhau.



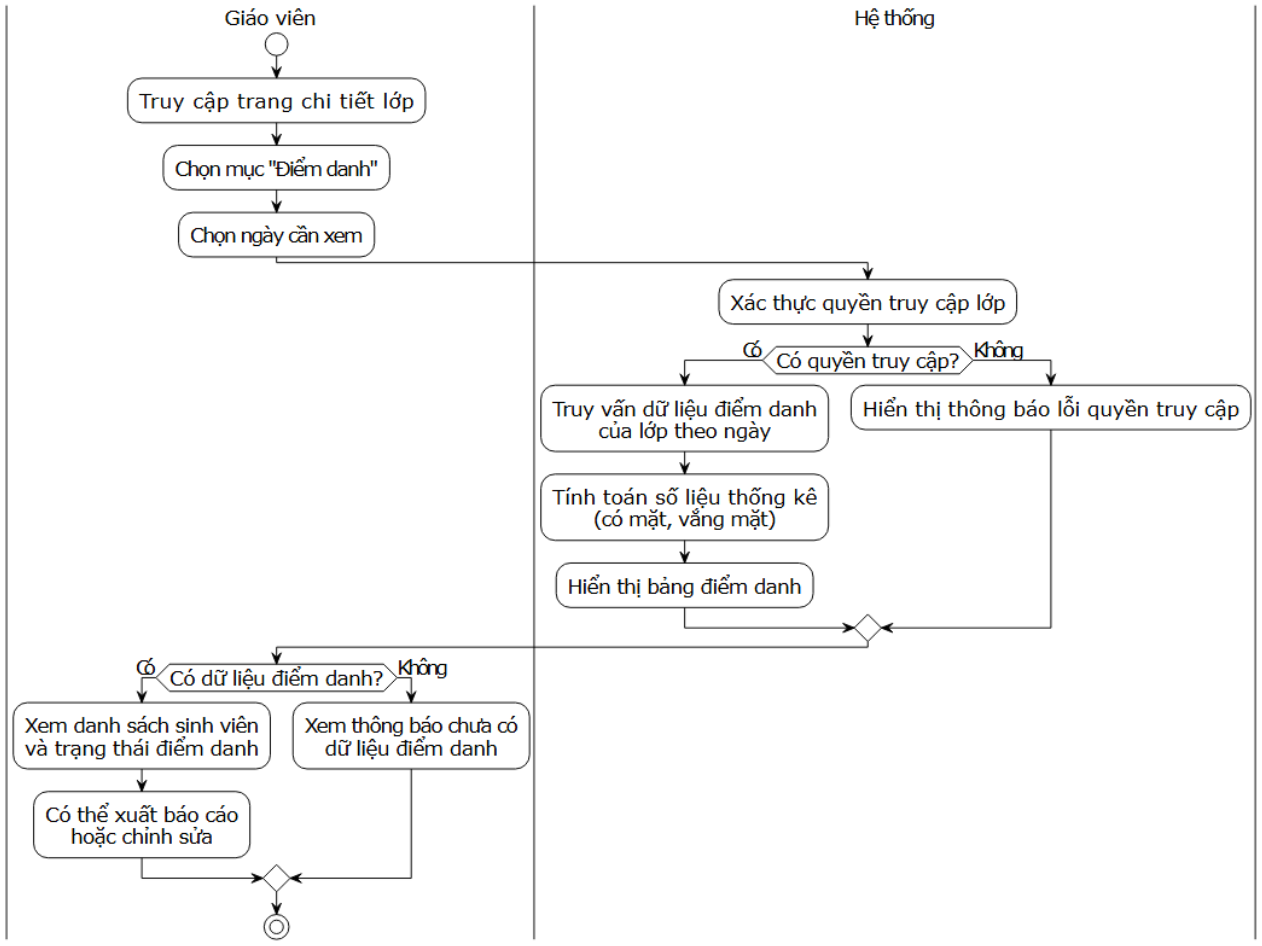
Hình 3.13: Biểu đồ hoạt động giáo viên xem chi tiết lớp

Quy trình thêm sinh viên vào lớp học bắt đầu khi giáo viên đã đăng nhập vào hệ thống và truy cập trang chi tiết của một lớp học cụ thể. Tại đây, giáo viên chọn chức năng "Thêm sinh viên" từ các tùy chọn quản lý lớp học. Hệ thống hiển thị một biểu mẫu cho phép giáo viên nhập thông tin của sinh viên mới, bao gồm họ tên và thông tin liên hệ (nếu cần), sau đó giáo viên nhấn nút xác nhận để gửi thông tin.

Sau khi nhận được yêu cầu, hệ thống kiểm tra quyền quản lý lớp học của giáo

viên hiện tại, đảm bảo rằng chỉ giáo viên chủ quản mới có thể thêm sinh viên vào lớp học của mình. Tiếp theo, hệ thống xử lý dữ liệu đầu vào, kiểm tra tính hợp lệ và tạo một bản ghi sinh viên mới trong cơ sở dữ liệu. Sinh viên này được liên kết với lớp học thông qua mã số lớp, tạo mối quan hệ trong hệ thống quản lý dữ liệu để phục vụ cho các chức năng điểm danh sau này.

Khi quá trình lưu trữ hoàn tất, hệ thống hiển thị thông báo thành công và cập nhật danh sách sinh viên trong trang chi tiết lớp học. Giáo viên có thể tiếp tục thêm nhiều sinh viên khác hoặc quay lại các chức năng quản lý lớp học. Từ thời điểm này, sinh viên mới được thêm vào có thể được nhận diện trong hệ thống điểm danh thông qua mã số sinh viên, nhưng cần hoàn tất việc đăng ký khuôn mặt để có thể sử dụng chức năng nhận diện khuôn mặt trong quá trình điểm danh.



Hình 3.14: Biểu đồ hoạt động giáo viên xem bản ghi điểm danh của lớp

Quy trình xem bản ghi điểm danh của lớp bắt đầu khi giáo viên đã đăng nhập vào hệ thống và truy cập trang chi tiết của một lớp học cụ thể. Tại đây, giáo viên chọn chức năng "Xem điểm danh" hoặc tab "Điểm danh" từ giao diện quản lý lớp học. Hệ thống cho phép giáo viên chọn ngày cần xem hoặc mặc định hiển thị bản ghi điểm danh của

ngày hiện tại, tùy thuộc vào thiết kế giao diện của hệ thống.

Khi giáo viên xác nhận lựa chọn ngày, hệ thống kiểm tra quyền truy cập của giáo viên đối với lớp học đó và tiến hành truy vấn cơ sở dữ liệu. Quá trình này bao gồm việc lấy danh sách tất cả học sinh trong lớp và tình trạng điểm danh tương ứng của từng học sinh (có mặt, vắng mặt, chưa điểm danh) cho ngày đã chọn. Hệ thống cũng tính toán các thông số thống kê như tổng số học sinh có mặt, tỷ lệ điểm danh, và các chỉ số liên quan khác để cung cấp cái nhìn tổng quan về tình hình tham gia học tập.

Kết quả truy vấn được hiển thị cho giáo viên dưới dạng bảng điểm danh, trong đó liệt kê danh sách học sinh kèm theo trạng thái điểm danh và thời gian điểm danh (nếu có). Giáo viên có thể thấy rõ những học sinh đã điểm danh, vắng mặt, hoặc chưa có thông tin. Từ giao diện này, giáo viên có thể thực hiện các thao tác bổ sung như xuất dữ liệu điểm danh ra file, chỉnh sửa trạng thái điểm danh của học sinh cụ thể, hoặc chuyển đổi sang xem dữ liệu điểm danh của những ngày khác. Chức năng này giúp giáo viên theo dõi hiệu quả sự tham gia của học sinh và đưa ra các quyết định giảng dạy phù hợp.

3.3. Kết chương:

Chương 3 đã hoàn thành nhiệm vụ quan trọng là phân tích và đặc tả chi tiết các yêu cầu của hệ thống điểm danh bằng nhận diện khuôn mặt. Thông qua việc xác định rõ ràng các chức năng dành cho từng tác nhân (Giáo viên và Học sinh), chương đã phác thảo nên bức tranh toàn diện về những gì người dùng mong đợi từ hệ thống. Các mô tả chức năng chi tiết đã làm rõ nghiệp vụ quản lý lớp học, đăng ký, điểm danh và theo dõi của cả hai đối tượng người dùng.

Việc sử dụng các biểu đồ Use Case đã giúp hệ thống hóa và trực quan hóa các tương tác chính giữa người dùng và hệ thống, cung cấp cái nhìn tổng quan về phạm vi và các quy trình cốt lõi. Quan trọng hơn, các biểu đồ hoạt động đã đi sâu vào mô tả luồng xử lý từng bước cho các ca sử dụng quan trọng, từ đăng ký tài khoản, quản lý lớp học, tải ảnh khuôn mặt cho đến thực hiện điểm danh và xem lại lịch sử. Các biểu đồ này không chỉ làm rõ logic hoạt động mà còn là tài liệu tham khảo hữu ích cho việc thiết kế và kiểm thử sau này.

Chương 4. PHÂN TÍCH THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG

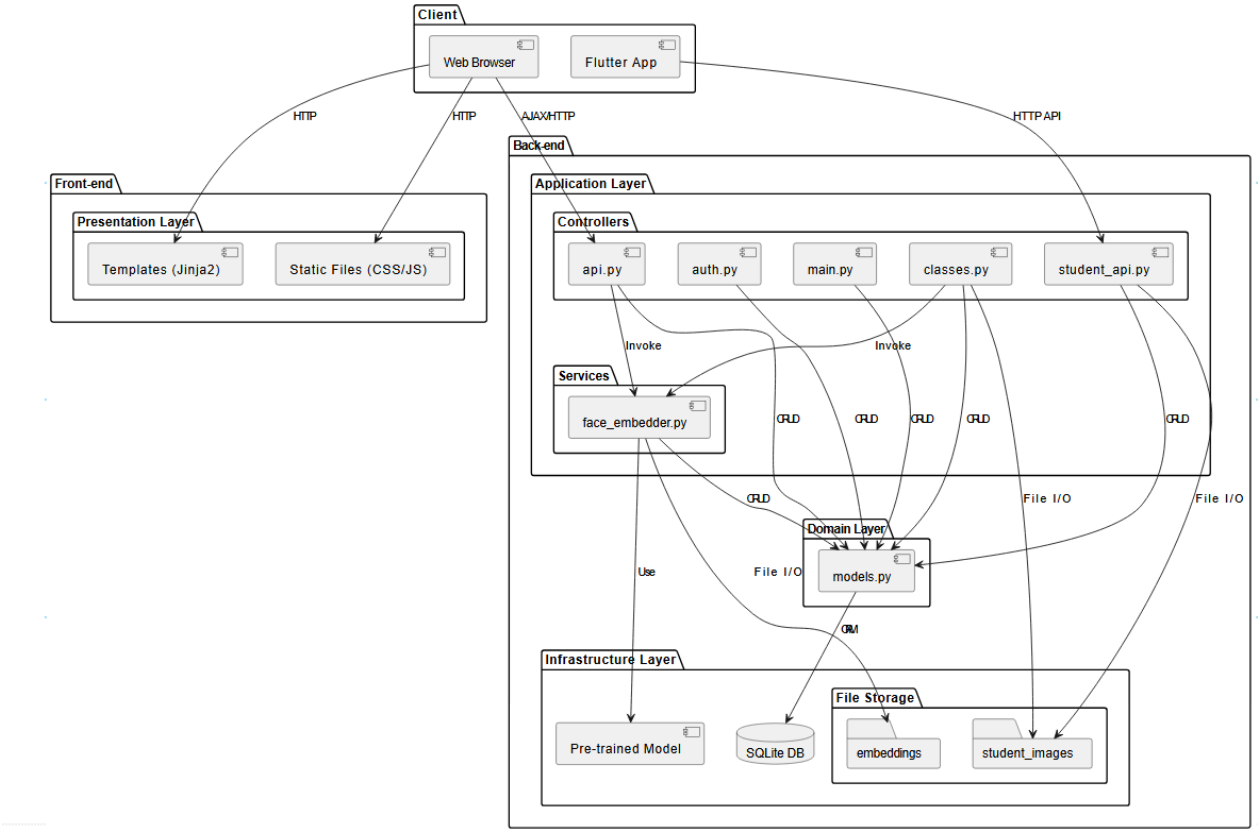
Chương này đóng vai trò chuyển đổi các yêu cầu và khái niệm thành cấu trúc kỹ thuật cụ thể của hệ thống điểm danh bằng nhận diện khuôn mặt. Mục tiêu là mô tả chi tiết kiến trúc hệ thống, cách thức tổ chức dữ liệu, giao diện lập trình ứng dụng (API), cơ chế hoạt động của module nhận diện cốt lõi và thiết kế giao diện người dùng, qua đó làm rõ cách thức hệ thống được xây dựng để đáp ứng các mục tiêu đề ra. Nội dung chương sẽ bắt đầu với việc trình bày kiến trúc tổng thể của hệ thống theo mô hình Client-Server. Tiếp theo, chương sẽ phân tích chi tiết thiết kế kiến trúc cho từng thành phần: Front-End Web dành cho giáo viên sử dụng Flask và Jinja2, Front-End ứng dụng di động cho sinh viên được phát triển bằng Flutter, và kiến trúc Back-end phân lớp dựa trên Flask.

Sau đó, thiết kế cơ sở dữ liệu quan hệ với cấu trúc các bảng chính sẽ được mô tả cụ thể, nhằm đảm bảo việc lưu trữ và truy xuất dữ liệu được thực hiện một cách khoa học và hiệu quả. Chương cũng sẽ định nghĩa các API endpoints cần thiết cho việc giao tiếp giữa client và server, tạo nên một cầu nối vững chắc và linh hoạt cho việc trao đổi thông tin và thực thi các yêu cầu nghiệp vụ. Phần quan trọng tiếp theo là phân tích và thiết kế Module nhận diện khuôn mặt, làm rõ quy trình xử lý từ phát hiện bằng MTCNN đến so khớp bằng FaceNet, đây là trái tim công nghệ của toàn bộ hệ thống.

Cuối cùng, chương sẽ đề cập đến các khía cạnh trong thiết kế giao diện người dùng, tập trung vào việc tạo ra một trải nghiệm trực quan, dễ sử dụng và thân thiện cho cả giáo viên và sinh viên. Việc trình bày một cách hệ thống và chi tiết các khía cạnh thiết kế và triển khai này không chỉ giúp hình dung rõ ràng về cấu trúc và hoạt động của hệ thống mà còn tạo cơ sở vững chắc cho việc đánh giá hiệu quả và khả năng ứng dụng của hệ thống ở các chương tiếp theo.

4.1. Kiến trúc hệ thống

4.1.1. Tổng quan kiến trúc hệ thống



Hình 4.1: Tổng quan kiến trúc hệ thống Client-Server

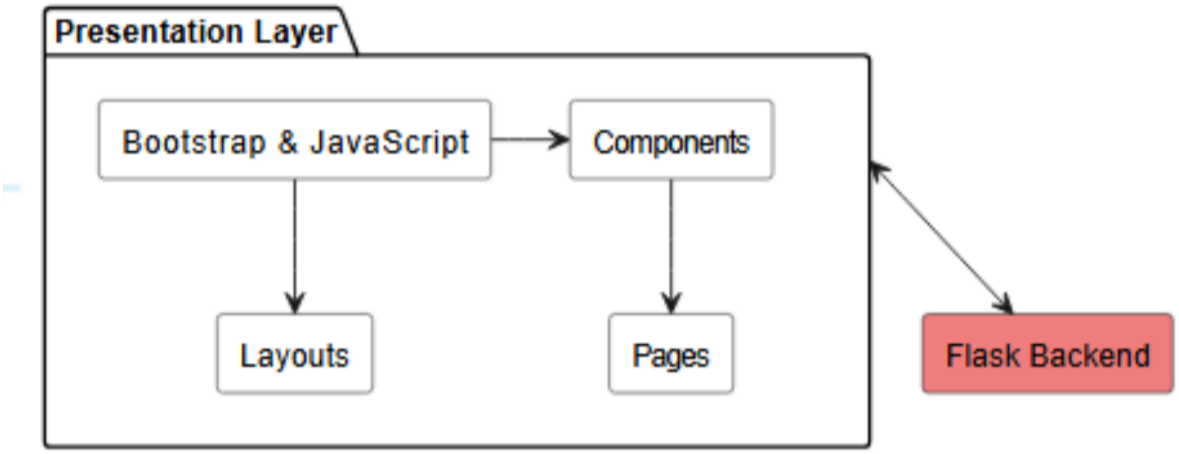
Mô hình hệ thống được thiết kế theo kiến trúc Client-Server, trong đó client (giao diện người dùng) gửi yêu cầu đến máy chủ backend để xử lý logic nghiệp vụ và truy xuất dữ liệu.

Bảng 4.1: Mô tả các thành phần chính trong kiến trúc tổng quan

STT	Thành phần	Mô tả
1	Dịch vụ Máy khách - Máy chủ (Client-Service)	Mô hình mạng trong đó nhiều máy khách kết nối và gửi yêu cầu dịch vụ đến một máy chủ trung tâm. Hệ thống bao gồm hai loại client: ứng dụng web cho giáo viên và ứng dụng Flutter cho sinh viên.
2	Kiến trúc Front-end	Bao gồm hai thành phần chính: (1) Web application sử dụng Jinja2 templates, Bootstrap CSS và JavaScript cho giáo viên; (2) Ứng dụng di động Flutter cho sinh viên để tương tác với hệ thống điểm danh.

3	Kiến trúc Back-end	Được xây dựng bằng Flask framework, tổ chức thành các Blueprint (auth, main, classes, api, student_api) để xử lý các yêu cầu HTTP. Back-end sử dụng SQLAlchemy để tương tác với cơ sở dữ liệu SQLite và face_embedder để xử lý nhận dạng khuôn mặt.

4.1.2. Thiết kế kiến trúc Front-End Web



Hình 4.2: Kiến trúc Front-end Web

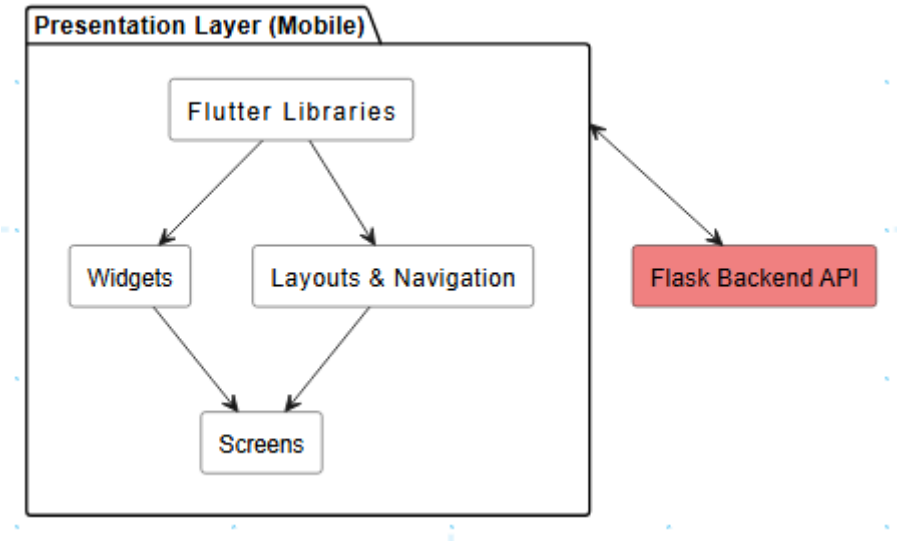
Kiến trúc front-end web của dự án được xây dựng dựa trên framework Flask phía backend, sử dụng Jinja2 làm template engine để render giao diện người dùng. Phần giao diện được cấu trúc thành các thành phần chính bao gồm Pages (trang riêng lẻ), Layouts (cấu trúc chung), và Components (thành phần tái sử dụng như form). Bootstrap và JavaScript (bao gồm jQuery) được sử dụng làm nền tảng CSS và xử lý các tương tác phía client, giao tiếp với Flask Backend để xử lý logic và dữ liệu.

Bảng 4.2: Mô tả các thành phần kiến trúc Front-End Web

STT	Thành phần	Mô tả
1	Presentation Layer	Lớp này bao gồm các thành phần của giao diện người dùng web (front-end) được xây dựng bằng Flask và render thông qua Jinja2 templates.
2	Pages	Chứa các tệp template Jinja2 (.html trong thư mục templates) tạo nên giao diện cho từng trang riêng lẻ. Các route trong Flask (routes) chịu trách nhiệm xử lý logic và render các template này tương ứng với yêu cầu của người dùng.

3	Layouts	Template base.html cung cấp khung tổng thể và cấu trúc chung (ví dụ: thanh điều hướng, vùng nội dung chính) được kế thừa bởi các trang khác. Giúp tạo sự nhất quán về giao diện và trải nghiệm người dùng trên toàn bộ ứng dụng web.
4	Component	Các phần tử hoặc mô-đun giao diện người dùng được tái sử dụng, chủ yếu bao gồm: Các Form được định nghĩa bằng Flask-WTF và được render thành các thẻ HTML , các cấu trúc HTML phổ biến như bảng (<table>), thông báo
5	Bootstrap & JavaScript Libraries	Sử dụng Bootstrap làm CSS framework chính để tạo kiểu và bố cục giao diện. JavaScript (bao gồm jQuery và mã tùy chỉnh trong script.js) được dùng để xử lý các tương tác phía client, ví dụ như gửi yêu cầu AJAX trong trang điểm danh.

4.1.3. Thiết kế kiến trúc Front-End ứng dụng di động



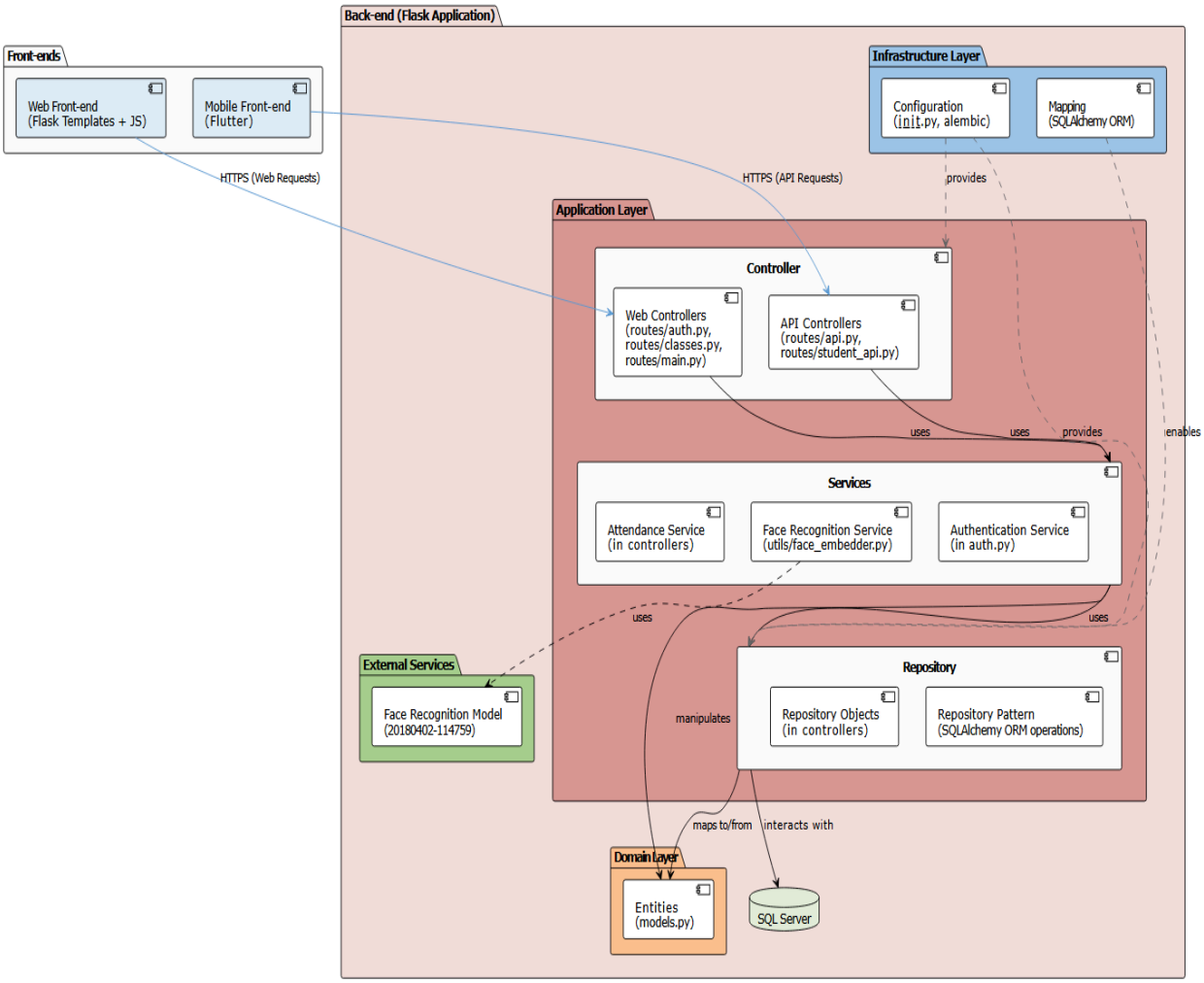
Bảng 4.2: Mô tả các thành phần kiến trúc Front-End Web

Kiến trúc front-end cho ứng dụng di động được phát triển bằng Flutter. Nền tảng này cung cấp các thư viện (Flutter Libraries) làm cơ sở để xây dựng giao diện người dùng. Giao diện được tổ chức thành các Screens (màn hình), sử dụng Widgets (các thành phần giao diện cơ bản và tùy chỉnh) và Layouts & Navigation để sắp xếp bố cục và quản lý luồng di chuyển giữa các màn hình. Ứng dụng di động này tương tác trực tiếp với Flask Backend API để trao đổi dữ liệu và thực hiện các chức năng nghiệp vụ.

Bảng 4.3: Mô tả các thành phần kiến trúc Front-End ứng dụng di động

STT	Thành phần	Mô tả
1	Flutter Libraries	Cung cấp các bộ thư viện và công cụ cốt lõi của Flutter để xây dựng giao diện người dùng và quản lý trạng thái ứng dụng.
2	Widgets	Là các khối xây dựng cơ bản của giao diện người dùng trong Flutter. Mọi thứ hiển thị trên màn hình đều là Widget (ví dụ: nút bấm, văn bản, hình ảnh).
3	Layouts & Navigation	Chịu trách nhiệm sắp xếp các Widgets trên màn hình (bố cục) và quản lý việc chuyển đổi giữa các màn hình (Screens) khác nhau trong ứng dụng.
4	Screens	Đại diện cho các màn hình giao diện hoàn chỉnh mà người dùng nhìn thấy và tương tác (ví dụ: màn hình đăng nhập, màn hình dashboard, màn hình điểm danh).
5	Flask Backend API	Giao diện lập trình ứng dụng (API) phía backend được xây dựng bằng Flask, nơi ứng dụng di động gửi yêu cầu và nhận dữ liệu.

4.1.4. Thiết kế kiến trúc Back-end



Hình 4.4: Kiến trúc Back-end

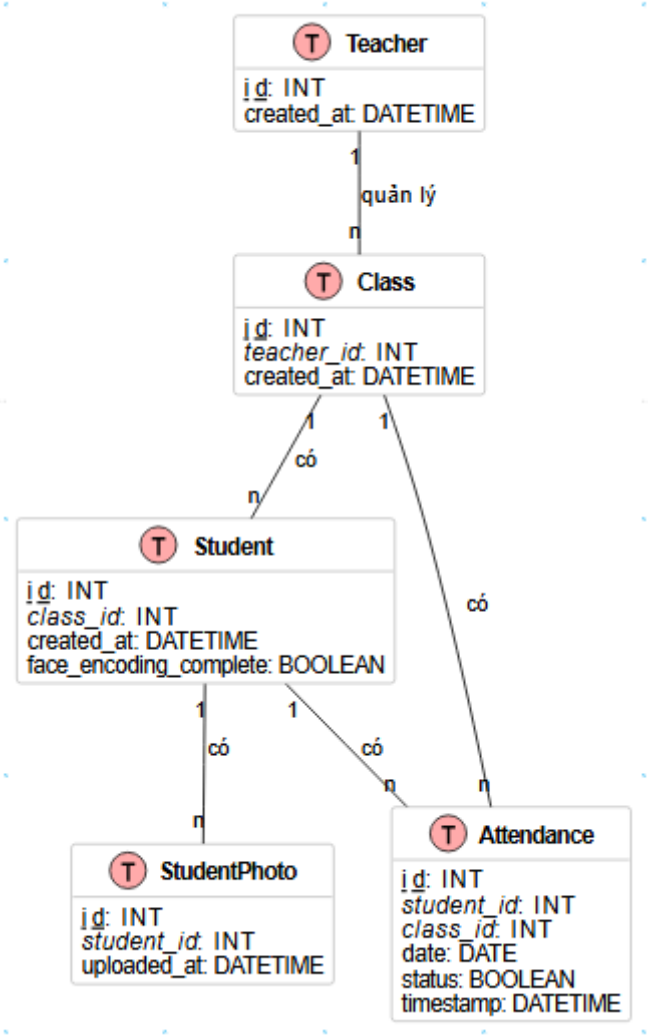
Kiến trúc Back-end của hệ thống điểm danh được xây dựng trên nền tảng Flask với cấu trúc phân lớp rõ ràng. Application Layer chứa các controllers riêng biệt cho web và mobile, các services xử lý nghiệp vụ như xác thực và nhận dạng khuôn mặt, cùng với repository quản lý dữ liệu qua SQLAlchemy. Domain Layer định nghĩa các entities cốt lõi trong models.py, trong khi Infrastructure Layer cung cấp cấu hình và cơ chế mapping ORM. Hệ thống tích hợp mô hình nhận dạng khuôn mặt như một dịch vụ bên ngoài, tạo nên một kiến trúc module hóa, dễ bảo trì và cung cấp API endpoints nhất quán cho cả giao diện web và ứng dụng di động. Sau đây là bảng thành phần mô tả kiến trúc

Bảng 4.4: Mô tả các thành phần kiến trúc Back-end

STT	Thành phần	Mô tả
1	Application Layer	Chứa các controllers (auth.py, classes.py, main.py, api.py, student_api.py) xử lý các yêu cầu từ front-end, các services như xác thực và nhận dạng khuôn mặt, cùng repository quản lý dữ liệu.
2	Domain Layer	Định nghĩa các entities cốt lõi của hệ thống trong models.py như Student, Class, và Attendance, thể hiện các đối tượng nghiệp vụ của ứng dụng.
3	Infrastructure Layer	Cung cấp cấu hình hệ thống thông qua init.py, alembic và cơ chế mapping của SQLAlchemy ORM để chuyển đổi giữa đối tượng và dữ liệu quan hệ.
4	External Services	Tích hợp mô hình nhận dạng khuôn mặt (Face Recognition Model) từ thư mục 20180402-114759 để thực hiện chức năng xác thực người dùng khi điểm danh.
5	Flask Backend API	Giao diện lập trình ứng dụng (API) phía backend được xây dựng bằng Flask, xử lý các yêu cầu từ cả giao diện web và ứng dụng di động, cung cấp endpoints cho các thao tác CRUD.

4.2. Thiết kế cơ sở dữ liệu

Hình 4.5: Sơ đồ quan hệ thực thể của cơ sở dữ liệu của hệ thống



Mô hình cơ sở dữ liệu của hệ thống điểm danh sử dụng nhận diện khuôn mặt được thiết kế theo cấu trúc quan hệ, với 5 bảng chính có mối quan hệ chặt chẽ với nhau. Mô hình này phản ánh đúng cấu trúc nghiệp vụ của một hệ thống quản lý điểm danh trong môi trường giáo dục, trong đó giáo viên quản lý các lớp học, mỗi lớp học có nhiều học sinh, và hoạt động điểm danh được thực hiện thông qua công nghệ nhận diện khuôn mặt. Thiết kế này hỗ trợ cả giao diện web cho giáo viên và ứng dụng di động cho học sinh, đảm bảo dữ liệu được lưu trữ và truy xuất hiệu quả.

4.2.1. Thành phần cơ sở dữ liệu bảng giáo viên:

Bảng 4.5: Cấu trúc bảng giáo viên

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của giáo viên, được tạo tự động và tăng dần
username	VARCHAR(64)	UNIQUE, NOT NULL	Tên đăng nhập của giáo viên, độc nhất trong hệ thống và không được phép trống
email	VARCHAR(120)	UNIQUE, NOT NULL	Địa chỉ email của giáo viên, dùng để liên lạc và đăng nhập, phải là duy nhất
password_hash	VARCHAR(128)		Mật khẩu đã được mã hóa, không lưu mật khẩu gốc
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Thời điểm tạo tài khoản giáo viên, mặc định là thời gian hiện tại

4.2.2. Thành phần cơ sở dữ liệu bảng lớp học:

Bảng 4.6: Cấu trúc bảng lớp học

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của lớp học, được tạo tự động và tăng dần
username	VARCHAR(100)	NOT NULL	Tên lớp học, bắt buộc phải có
teacher_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của giáo viên phụ trách lớp, liên kết với bảng Teacher
created_at	DATETIME	DEFAULT CURRENT_TI	Thời điểm tạo lớp học, mặc định là thời gian hiện tại

		MESTAMP	
class_code	VARCHAR(8)	UNIQUE, NOT NULL	Mã lớp học dùng cho học sinh tham gia, là duy nhất

4.2.3. Thành phần cơ sở dữ liệu bảng học sinh:

Bảng 4.7: Cấu trúc bảng học sinh

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của học sinh, được tạo tự động và tăng dần
name	VARCHAR (100)	NOT NULL	Tên học sinh, bắt buộc phải có
email	VARCHAR (120)	UNIQUE	Địa chỉ email của học sinh, dùng để liên lạc và đăng nhập vào ứng dụng di động
password_hash	VARCHAR (128)		Mật khẩu đã được mã hóa, dùng để đăng nhập vào ứng dụng di động
class_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của lớp học mà học sinh tham gia, liên kết với bảng Class
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Thời điểm học sinh được thêm vào hệ thống
face_encoding_complete	BOOLEAN	DEFAULT FALSE	Cờ đánh dấu trạng thái hoàn thành mã hóa khuôn mặt, mặc định là False (chưa hoàn thành)

4.2.4. Thành phần cơ sở dữ liệu bảng ảnh học sinh:

Bảng 4.8: Cấu trúc bảng ảnh học sinh

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của bản ghi ảnh, được tạo tự động và tăng dần
filename	VARCHAR(255)	NOT NULL	Đường dẫn đến tệp ảnh khuôn mặt của học sinh
student_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của học sinh sở hữu ảnh, liên kết với bảng Student
uploaded_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Thời điểm ảnh được tải lên hệ thống

4.2.4. Thành phần cơ sở dữ liệu bảng điểm danh:

Bảng 4.9: Cấu trúc bảng điểm danh

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của bản ghi điểm danh, được tạo tự động và tăng dần
student_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của học sinh được điểm danh, liên kết với bảng Student
class_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của lớp học trong buổi điểm danh, liên kết với bảng Class
date	DATE	NOT NULL	Ngày diễn ra buổi học được điểm danh
status	BOOLEAN	DEFAULT FALSE	Trạng thái điểm danh: True = Có mặt, False = Vắng mặt

Trường dữ liệu	Kiểu dữ liệu	Ràng buộc	Mô tả
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Mã định danh duy nhất của bản ghi điểm danh, được tạo tự động và tăng dần
student_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của học sinh được điểm danh, liên kết với bảng Student
class_id	INTEGER	FOREIGN KEY, NOT NULL	Mã định danh của lớp học trong buổi điểm danh, liên kết với bảng Class
date	DATE	NOT NULL	Ngày diễn ra buổi học được điểm danh
timestamp	DATETIME	DEFAULT CURRENT_TIMESTAMP	Thời điểm chính xác khi điểm danh được thực hiện
face_encoding_complete	BOOLEAN	DEFAULT FALSE	Cờ đánh dấu trạng thái hoàn thành mã hóa khuôn mặt, mặc định là False (chưa hoàn thành)

4.3. Thiết kế API

4.3.1. API Quản lý học sinh

Bảng 4.10: Danh sách API quản lý học sinh

STT	Đường dẫn	Phương thức	Chức năng
1	/api/student/register	POST	Đăng ký tài khoản học sinh mới
2	/api/student/login	POST	Đăng nhập và nhận thông tin học sinh
3	/api/student/join_class	POST	Tham gia vào lớp học bằng mã lớp
4	/api/student/upload_faces	POST	Tải lên ảnh khuôn mặt của học sinh
5	/api/student/submit_attendance	POST	Điểm danh bằng nhận diện khuôn mặt

6	/api/student/attendance_history/{student_id}	GET	Lấy lịch sử điểm danh của học sinh
7	/api/student/profile/{student_id}	GET	Lấy thông tin cá nhân của học sinh

4.3.2. API điểm danh

Bảng 4.11: Danh sách API điểm danh

STT	Đường dẫn	Phương thức	Chức năng
1	/api/recognize	POST	Nhận diện khuôn mặt từ ảnh và xác định học sinh
2	/api/save-attendance	POST	Lưu kết quả điểm danh cho lớp học
3	/check-face	POST	Kiểm tra xem có nhận diện được khuôn mặt trong ảnh không
4	/classes/{class_id}/attendance-data	GET	Lấy dữ liệu điểm danh cho một lớp học cụ thể
5	/api/get-attendance-data	GET	Lấy dữ liệu điểm danh học sinh theo lớp và ngày

4.3.3. API xác thực

Bảng 4.11: Danh sách API điểm danh

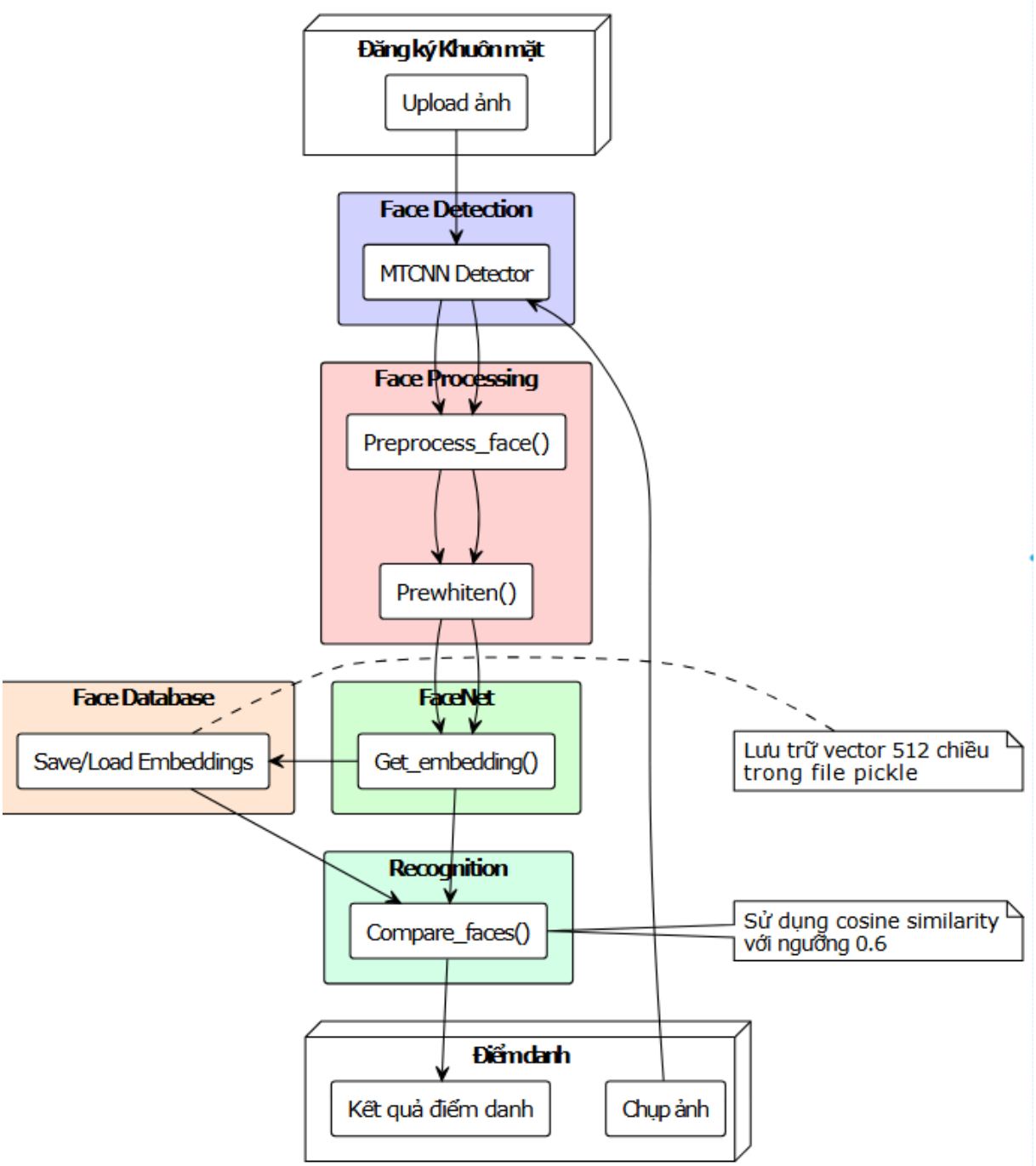
STT	Đường dẫn	Phương thức	Chức năng
1	/api/student/login	POST	Đăng nhập và nhận token xác thực
2	/api/student/register	POST	Đăng ký tài khoản người dùng mớ

4.3.4. API lấy thông tin học sinh

Bảng 4.13: Danh sách API lấy thông tin học sinh

STT	Đường dẫn	Phương thức	Chức năng
1	/api/student/join_class	POST	Tham gia vào lớp học bằng mã lớp
2	/api/student/upload_faces	POST	Tải lên ảnh khuôn mặt của học sinh
3	/api/student/profile/{student_id}	GET	Lấy thông tin cá nhân của học sinh
4	/api/student/attendance_history/{student_id}	GET	Lấy lịch sử điểm danh của học sinh

4.4. Thiết kế Module nhận diện khuôn mặt



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

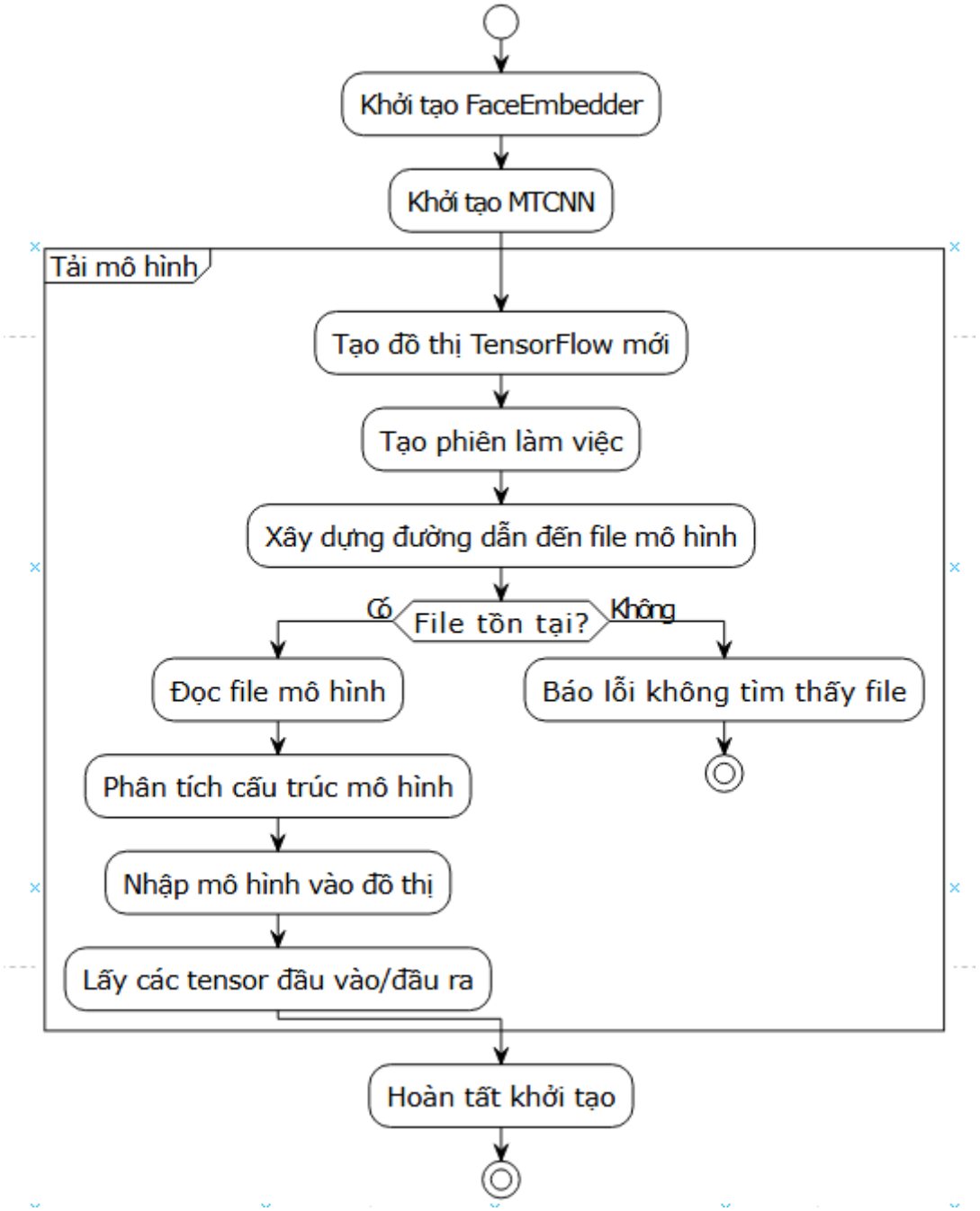
Quy trình bắt đầu tại tầng phát hiện khuôn mặt (Face Detection), nơi bộ phát hiện MTCNN được sử dụng để xác định vị trí và kích thước của khuôn mặt trong ảnh đầu vào. MTCNN đặc biệt hiệu quả nhờ kiến trúc mạng cascade ba tầng, cho phép phát hiện khuôn mặt ở nhiều tỷ lệ khác nhau với độ chính xác cao. Khi khuôn mặt được phát hiện, hệ thống trích xuất vùng khuôn mặt cùng với một lề 20% để đảm bảo

không bỏ sót các đặc điểm quan trọng ở rìa khuôn mặt.

Tiếp theo, quy trình chuyển sang giai đoạn xử lý khuôn mặt (Face Processing) với hai bước quan trọng: preprocess_face() và prewhiten(). Phương thức preprocess_face() chuẩn hóa kích thước khuôn mặt về 160x160 pixels và chuyển đổi giá trị pixel sang khoảng [0,1]. Phương thức prewhiten() tiếp tục xử lý ảnh bằng cách chuẩn hóa độ tương phản thông qua việc trừ đi giá trị trung bình và chia cho độ lệch chuẩn đã điều chỉnh, giảm thiểu ảnh hưởng của điều kiện ánh sáng và tăng cường đặc trưng khuôn mặt.

Sau khi khuôn mặt đã được xử lý, nó được đưa vào mô hình FaceNet thông qua phương thức get_embedding(). FaceNet chuyển đổi ảnh khuôn mặt 160x160 pixels thành một vector đặc trưng (embedding) 512 chiều, đại diện cho các đặc điểm sinh trắc học của khuôn mặt. Vector này được chuẩn hóa về độ dài đơn vị và lưu trữ trong cơ sở dữ liệu dưới dạng file pickle. Khi cần xác thực danh tính, phương thức compare_faces() tính toán độ tương đồng cosine giữa vector embedding của khuôn mặt cần xác thực và các vector đã lưu trữ. Nếu độ tương đồng vượt qua ngưỡng 0.6, hệ thống xác nhận đó là cùng một người, cho phép quá trình điểm danh diễn ra chính xác và an toàn, ngăn chặn hiệu quả các trường hợp giả mạo danh tính trong hệ thống điểm danh.

4.4.1. Khởi tạo và tải mô hình FaceNet

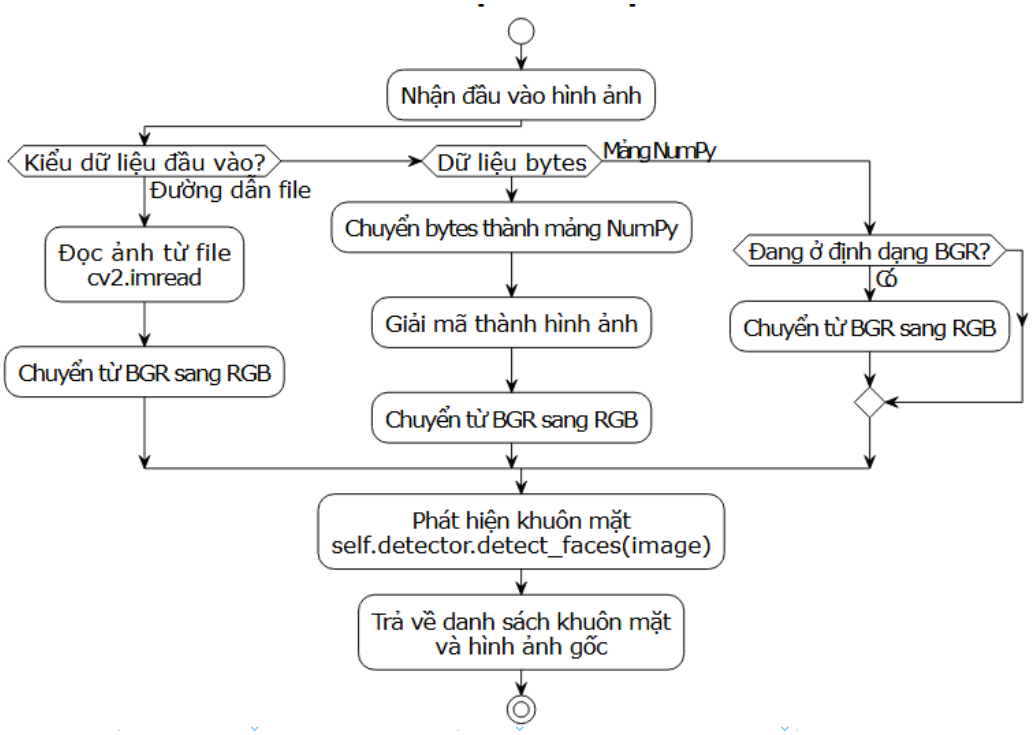


Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

FaceNet so sánh khuôn mặt trong hệ thống điểm danh của bạn thông qua phương thức compare_faces() được triển khai tinh tế. Phương thức này nhận vector embedding của khuôn mặt cần xác thực và từ điển chứa các embedding đã lưu trữ của học sinh. Quá trình so sánh dựa trên độ tương đồng cosine - được tính bằng tích vô hướng của hai vector đã chuẩn hóa (np.dot(embedding, stored_embedding)). Độ tương đồng này có giá trị từ -1 đến 1, trong đó giá trị càng gần 1 thể hiện hai khuôn mặt càng giống

nhau. Ngưỡng mặc định 0.6 đảm bảo cân bằng giữa độ chính xác và khả năng nhận diện - nếu độ tương đồng vượt ngưỡng này và cao hơn tất cả các giá trị khác, học sinh tương ứng được coi là khớp. Phương pháp này vừa hiệu quả về mặt tính toán vừa chính xác trong nhận diện, giúp hệ thống điểm danh hoạt động đáng tin cậy ngay cả khi có sự khác biệt nhỏ về góc chụp và ánh sáng.

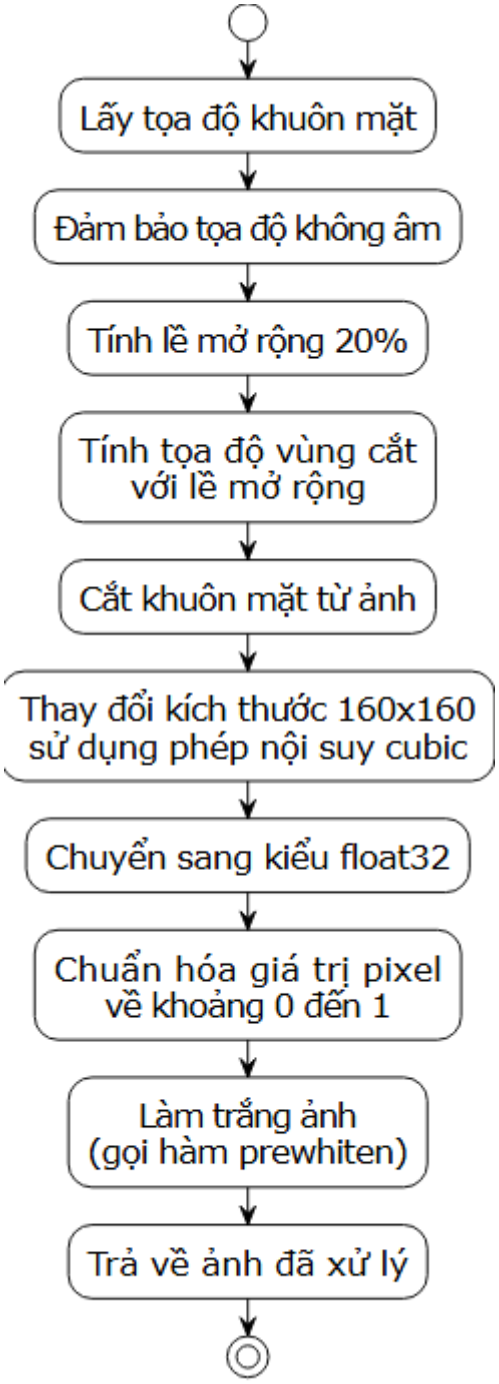
4.4.2. Phát hiện khuôn mặt sử dụng MTCNN



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

FaceNet phát hiện khuôn mặt trong hệ thống điểm danh của bạn thông qua phương thức detect_faces() được thiết kế linh hoạt. Phương thức này xử lý đa dạng định dạng đầu vào gồm đường dẫn file, dữ liệu byte, và mảng NumPy, cho phép tương thích với nhiều nguồn ảnh khác nhau. Ảnh đầu vào được đọc bằng OpenCV [3] thông qua cv2.imread() hoặc chuyển đổi từ dữ liệu byte bằng cv2.imdecode(), sau đó được chuyển từ không gian màu BGR sang RGB bằng cv2.cvtColor() để phù hợp với yêu cầu của MTCNN và FaceNet. Bộ phát hiện MTCNN sau đó được gọi để phân tích ảnh và trả về danh sách các khuôn mặt với thông tin chi tiết bao gồm tọa độ khung hình, độ tin cậy và các điểm mốc (landmarks) như mắt, mũi và miệng. Mỗi khuôn mặt được phát hiện đều có thông tin phong phú giúp cho các bước xử lý tiếp theo như căn chỉnh và nhận diện trở nên chính xác hơn, đặc biệt trong các điều kiện chụp ảnh phức tạp.

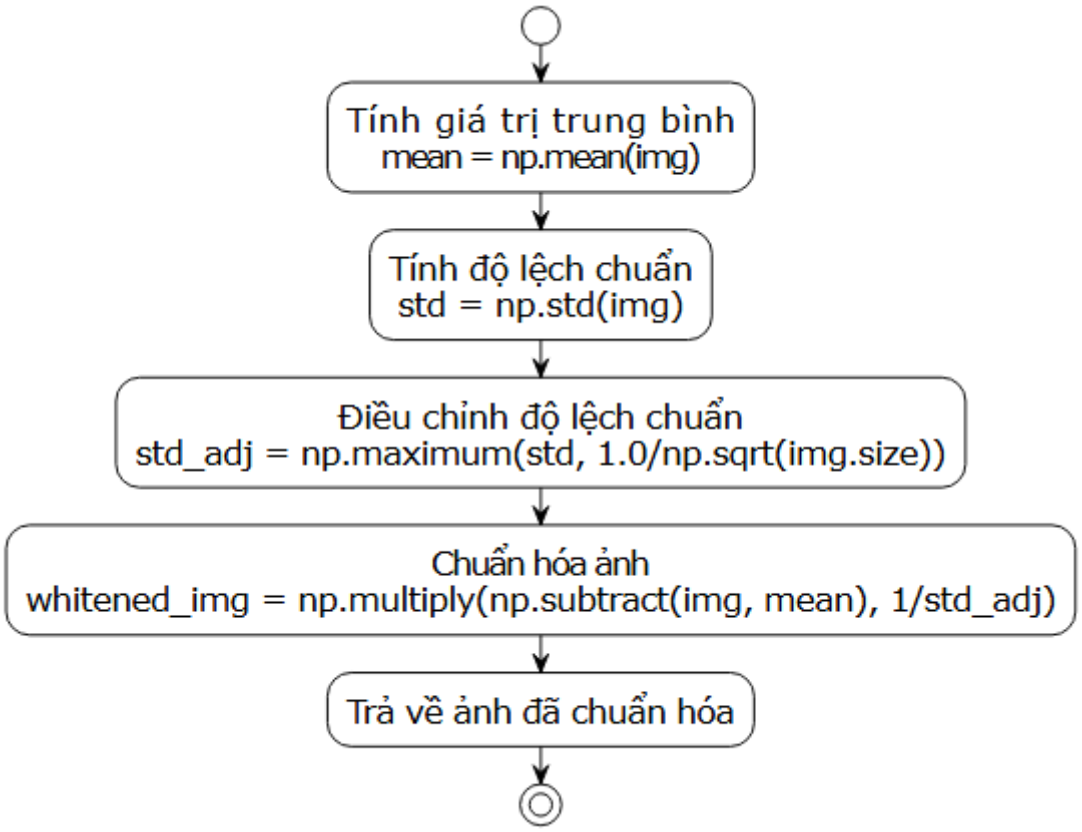
4.4.3. Tiền xử lý khuôn mặt



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

FaceNet tiền xử lý khuôn mặt trong hệ thống điểm danh của bạn thông qua phương thức preprocess_face() với quy trình nhiều bước tinh vi. Sau khi MTCNN phát hiện khuôn mặt, phương thức này trích xuất vùng khuôn mặt từ ảnh gốc với lề mở rộng 20% để đảm bảo không bỏ sót các đặc điểm quan trọng ở viền. Khuôn mặt được cắt sau đó được thay đổi kích thước về chuẩn 160x160 pixels bằng phép nội suy cubic,

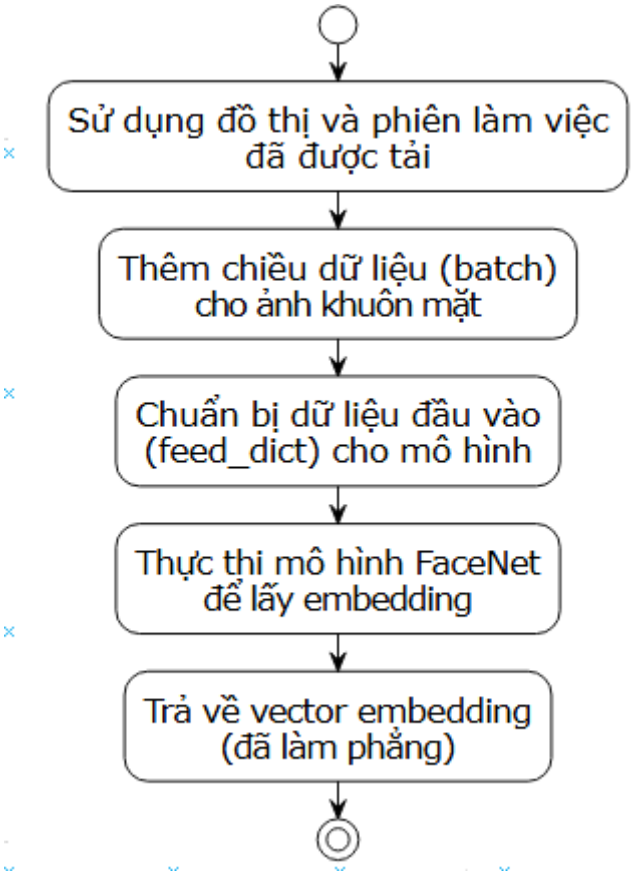
giữ lại nhiều chi tiết hơn so với các phương pháp thay đổi kích thước thông thường.



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

Tiếp theo, ảnh được chuyển đổi sang định dạng float32 và chuẩn hóa về khoảng [0,1] bằng cách chia cho 255. Bước cuối cùng là áp dụng kỹ thuật làm trắng ảnh của David Sandberg, tính toán giá trị trung bình và độ lệch chuẩn của ảnh, sau đó chuẩn hóa bằng cách trừ đi trung bình và chia cho độ lệch chuẩn đã điều chỉnh. Phương pháp tiền xử lý toàn diện này giúp loại bỏ sự thay đổi về ánh sáng và tăng cường đặc trưng khuôn mặt, cải thiện đáng kể độ chính xác của quá trình nhận dạng.

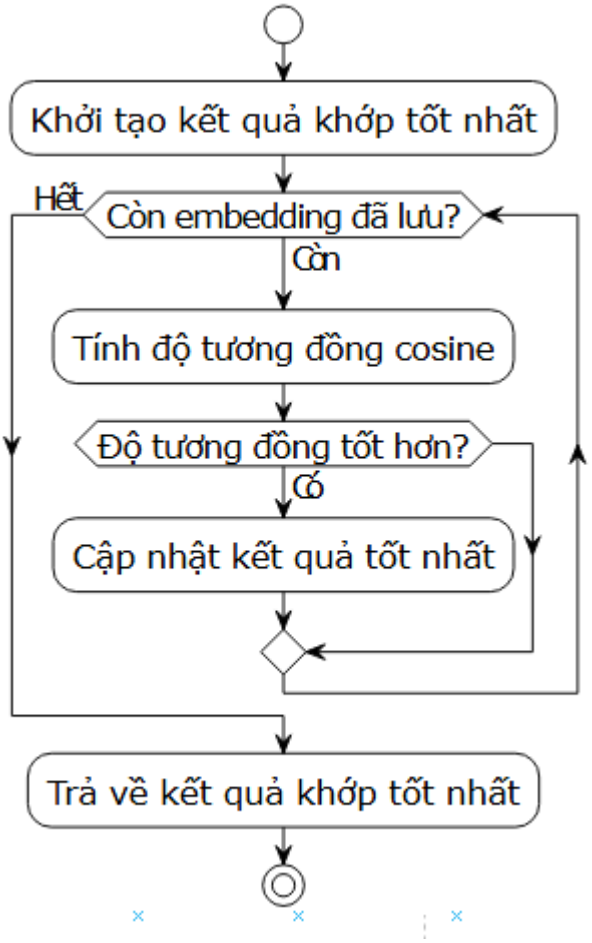
4.4.4. Tính toán embedding khuôn mặt



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

FaceNet tính toán face embedding trong hệ thống điểm danh của bạn thông qua phương thức get_embedding() - bước then chốt của toàn bộ quy trình. Sau khi khuôn mặt được phát hiện và tiền xử lý, phương thức này chịu trách nhiệm chuyển đổi ảnh khuôn mặt thành vector số học 512 chiều đại diện cho đặc trưng sinh trắc học. Đầu tiên, phương thức thêm một chiều batch vào tensor ảnh bằng np.expand_dims(), biến đổi từ kích thước (160, 160, 3) thành (1, 160, 160, 3) để phù hợp với đầu vào của mạng neural. Tiếp theo, một từ điển feed_dict được tạo ra với hai giá trị: ảnh đầu vào gán cho tensor images_placeholder và giá trị False gán cho phase_train_placeholder, chỉ định rằng mô hình đang hoạt động ở chế độ dự đoán, không phải đào tạo. Phương thức sau đó chạy mô hình FaceNet để tính toán vector embedding từ ảnh khuôn mặt. Kết quả là một vector 512 số thực - "chữ ký số" của khuôn mặt - với đặc tính quan trọng là khuôn mặt của cùng một người sẽ tạo ra các vector gần nhau trong không gian Euclidean, trong khi khuôn mặt của những người khác nhau sẽ tạo ra các vector cách xa nhau, làm nền tảng cho khả năng nhận diện chính xác của hệ thống.

4.4.5. So sánh khuôn mặt và tính toán sự tương đồng



Hình 4.6: Cơ chế hoạt động của Facenet trong hệ thống

FaceNet so sánh khuôn mặt trong hệ thống điểm danh của bạn thông qua phương thức compare_faces() với thuật toán hiệu quả và đơn giản. Tại cốt lõi của phương thức là việc sử dụng độ tương đồng cosine - được tính bằng tích vô hướng giữa vector embedding cần xác thực và các vector embedding đã được lưu trữ trong từ điển. Phương thức lặp qua từng cặp khóa-giá trị trong từ điển embeddings_dict, nơi mỗi khóa là ID của học sinh và mỗi giá trị là vector embedding đã chuẩn hóa tương ứng. Với mỗi học sinh, hàm tính toán độ tương đồng cosine và so sánh với ngưỡng (mặc định là 0.6). Nếu độ tương đồng vượt ngưỡng và lớn hơn tất cả các giá trị đã tìm thấy trước đó, hệ thống cập nhật best_match và best_similarity. Kết quả cuối cùng là một cặp giá trị - ID của học sinh khớp với độ tin cậy cao nhất và độ tương đồng tương ứng. Ngưỡng 0.6 này là điểm cân bằng tối ưu giữa độ chính xác và độ bao phủ, đảm bảo hệ thống đủ chặt chẽ để tránh nhận dạng nhầm nhưng vẫn đủ linh hoạt để thích ứng với các biến thể của khuôn mặt học sinh trong điều kiện chụp ảnh thực tế.

4.5. Thiết kế giao diện người dùng

Hệ thống điểm danh bằng nhận diện khuôn mặt được thiết kế với mục tiêu tạo ra một trải nghiệm người dùng trực quan, dễ sử dụng và hiệu quả. Ứng dụng được xây dựng dựa trên nền tảng Flutter, cho phép triển khai giao diện nhất quán trên các nền tảng di động khác nhau. Các nguyên tắc thiết kế Material Design của Google đã được áp dụng xuyên suốt quá trình phát triển, đảm bảo tính thống nhất và quen thuộc với người dùng.

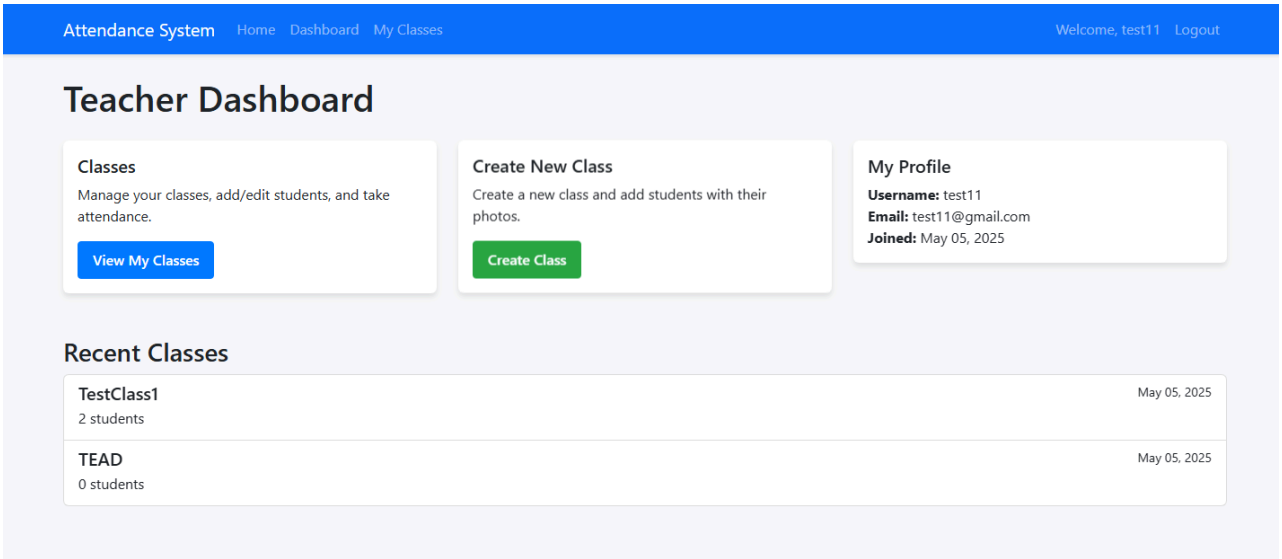
Triết lý thiết kế của hệ thống tập trung vào sự đơn giản và hiệu quả, tuân theo nguyên tắc "Material is the metaphor" - một trong những nguyên tắc cốt lõi của Material Design [7]. Với đối tượng người dùng chính là sinh viên, giao diện được tối ưu hóa để giảm thiểu số lượng thao tác cần thiết khi thực hiện các chức năng cốt lõi như điểm danh và xem lịch sử. Điều này đặc biệt quan trọng trong môi trường lớp học, nơi thời gian là yếu tố quan trọng và việc điểm danh cần được thực hiện nhanh chóng, chính xác.

Bảng màu của ứng dụng được chọn lựa dựa trên công cụ Material Color Tool [3], với màu xanh dương (Blue #2196F3) làm tông màu chủ đạo, tượng trưng cho sự chuyên nghiệp và đáng tin cậy, phù hợp với môi trường giáo dục. Màu trắng (#FFFFFF) được sử dụng làm nền chính để tăng khả năng đọc và tạo cảm giác thoáng đãng cho giao diện. Các màu phụ như xanh lá cây (#4CAF50) và đỏ (#F44336) được sử dụng có chọn lọc để biểu thị trạng thái thành công hoặc lỗi, tạo ra những điểm nhấn trực quan quan trọng theo nguyên tắc "Intentional color" của Material Design .

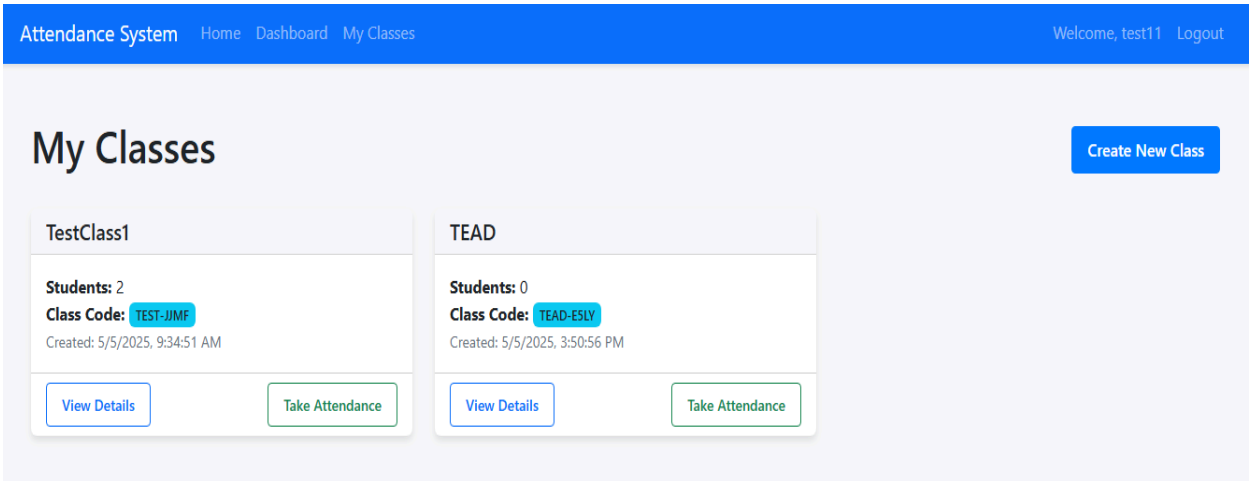
Typography trong ứng dụng tuân theo hệ thống phân cấp chữ của Material Design, sử dụng font Roboto - font chuẩn của Material Design, với các kích thước và trọng lượng khác nhau để tạo ra thứ bậc thông tin rõ ràng. Tiêu đề sử dụng Roboto Bold (24sp) với kích thước lớn hơn, trong khi nội dung chính sử dụng Roboto Regular (16sp) với kích thước vừa phải, đảm bảo tính dễ đọc trên màn hình nhỏ của thiết bị di động.

Các thành phần tương tác như nút bấm được thiết kế với kích thước tối thiểu 48x48dp để dễ dàng nhấn trên màn hình cảm ứng, tuân theo nguyên tắc "touch target" của Material Design. Thiết kế bố cục (layout) của ứng dụng tuân theo hệ thống lưới 8dp (8dp grid system) của Material Design, tạo ra sự nhất quán trong căn chỉnh, khoảng cách và kích thước các thành phần giao diện. Điều này giúp tạo ra một giao diện có tính thẩm mỹ cao và dễ sử dụng trên nhiều kích thước màn hình khác nhau.

Dưới đây là một số ví dụ về giao diện thiết kế trong hệ thống



Hình 4.7: Giao diện bảng giáo viên



Hình 4.8: Giao diện bảng các lớp học

4.6. Kết chương

Chương 4 đã trình bày chi tiết về quá trình phân tích thiết kế và các khía cạnh triển khai kỹ thuật của hệ thống điểm danh bằng nhận diện khuôn mặt. Kiến trúc tổng thể Client-Server đã được xác định, cùng với thiết kế cụ thể cho Front-End Web, Front-End Mobile, và Back-end, đảm bảo sự phân tách rõ ràng và khả năng tương tác hiệu quả giữa các thành phần. Thiết kế cơ sở dữ liệu quan hệ với các bảng chính đã được định nghĩa chi tiết, phản ánh cấu trúc nghiệp vụ và hỗ trợ lưu trữ, truy xuất dữ liệu.

Bộ API cần thiết cho việc quản lý học sinh, điểm danh, xác thực và truy xuất thông tin cũng đã được thiết kế, tạo cầu nối giao tiếp chuẩn hóa giữa client và server. Đặc biệt, chương đã đi sâu vào thiết kế Module nhận diện khuôn mặt, làm rõ quy trình xử lý ảnh bằng MTCNN và FaceNet, từ phát hiện, tiền xử lý, tính toán embedding đến so sánh độ tương đồng, đây là yếu tố công nghệ cốt lõi của hệ thống. Phần thiết kế giao diện người dùng, dù cần bổ sung chi tiết hình ảnh, cũng đã được xác định là một thành phần trong cấu trúc tổng thể.

Chương 5. KẾT QUẢ VÀ ĐÁNH GIÁ

Nội dung chính của chương sẽ bắt đầu bằng việc mô tả môi trường cài đặt cần thiết, bao gồm các yêu cầu về phần cứng, phần mềm, các thư viện và framework cụ thể được sử dụng cho cả phía Server (Backend) và Client (Ứng dụng di động). Tiếp theo, chương sẽ cung cấp một quy trình cài đặt chi tiết từng bước cho cả Backend và ứng dụng di động, giúp người dùng có thể tự triển khai hệ thống. Phần trọng tâm của chương là trình bày kết quả kiểm thử hệ thống một cách toàn diện. Điều này bao gồm việc kiểm thử Module Nhận diện Khuôn mặt để đánh giá độ chính xác và hiệu năng nhận diện, kiểm thử các API Backend để đảm bảo tính ổn định, bảo mật và đúng đắn của logic nghiệp vụ, và kiểm thử ứng dụng di động trên nhiều nền tảng để đánh giá tính năng, giao diện và trải nghiệm người dùng.

Các kết quả kiểm thử và đánh giá được trình bày trong chương này sẽ là bằng chứng thực nghiệm cho hiệu quả của giải pháp đã đề xuất và triển khai, làm cơ sở cho các kết luận cuối cùng về đề tài.

5.1. Môi trường cài đặt

5.1.1. Yêu cầu hệ thống

Hệ thống điểm danh dựa trên nhận diện khuôn mặt được phát triển với yêu cầu hệ thống như sau:

Đối với Server (Backend):

1. Hệ điều hành: Windows/Linux/macOS
2. Python 3.7+
3. Flask framework và các thư viện phụ thuộc
4. TensorFlow 2.x cho mô hình nhận diện khuôn mặt FaceNet
5. SQLAlchemy và SQL Server làm cơ sở dữ liệu
6. MTCNN cho phát hiện khuôn mặt
7. Ít nhất 4GB RAM và 1GB không gian lưu trữ

Đối với Client (Mobile App):

1. Flutter SDK 2.0+
2. Dart 2.12+
3. Android 6.0+ hoặc iOS 11.0+

4. Camera có độ phân giải tối thiểu 2MP

5.1.2. Các thư viện và framework sử dụng

Backend:

1. Flask: Web framework xây dựng API và giao diện web
2. TensorFlow: Nền tảng học máy cho mô hình FaceNet
3. MTCNN: Thuật toán phát hiện khuôn mặt
4. OpenCV: Xử lý và tiền xử lý hình ảnh
5. SQLAlchemy: ORM (Object-Relational Mapping) cho cơ sở dữ liệu
6. Flask-Login: Xác thực người dùng
7. Flask-Migrate: Quản lý phiên bản cơ sở dữ liệu

Frontend (Mobile):

1. Flutter: Framework UI đa nền tảng
2. http: Thư viện gửi request HTTP

5.2. Quy trình cài đặt

5.2.1. Cài đặt Backend

Để cài đặt môi trường Backend, trước tiên cần đảm bảo hệ thống đã có Python phiên bản 3.7 trở lên. Việc kiểm tra và nâng cấp pip, công cụ quản lý gói của Python, lên phiên bản mới nhất cũng được khuyến nghị để đảm bảo tương thích.

Tiếp theo, các thư viện Python cần thiết cho dự án, được liệt kê trong tệp requirements.txt, cần được cài đặt. Quá trình này thực hiện tự động bằng cách mở terminal, di chuyển vào thư mục gốc của dự án Backend và chạy lệnh pip install -r requirements.txt.

Mô hình FaceNet đã được huấn luyện trước cũng cần được tải về. Sau khi tải xong, tệp mô hình này phải được đặt vào đúng thư mục được chỉ định trong cấu hình của dự án để module nhận diện có thể truy cập.

Sau khi cài đặt thư viện và mô hình, bước tiếp theo là khởi tạo hoặc cập nhật cấu trúc cơ sở dữ liệu. Sử dụng công cụ Flask-Migrate, chạy lệnh flask db upgrade trong terminal tại thư mục gốc dự án để tạo hoặc áp dụng các thay đổi cần thiết cho các bảng dữ liệu.

Cuối cùng, để khởi động server Backend, chỉ cần thực thi tệp kịch bản chính của ứng dụng Flask (thường là run.py) bằng lệnh python run.py. Server sau đó sẽ lắng nghe và sẵn sàng xử lý các yêu cầu từ client.

5.2.2. Cài đặt ứng dụng di động

Việc cài đặt ứng dụng di động Flutter bắt đầu bằng việc tải và thiết lập Flutter SDK theo hướng dẫn chính thức, bao gồm cả Dart SDK và cấu hình môi trường phát triển phù hợp (Android Studio/SDK, Xcode). Công cụ flutter doctor nên được chạy để kiểm tra và hoàn tất các yêu cầu môi trường.

Sau khi Flutter SDK sẵn sàng, cần di chuyển vào thư mục gốc chứa mã nguồn của ứng dụng di động trong terminal và chạy lệnh flutter pub get. Lệnh này sẽ tải về tất cả các gói thư viện Dart và Flutter mà ứng dụng yêu cầu.

Một bước cấu hình quan trọng là cập nhật địa chỉ API Backend trong mã nguồn của ứng dụng Flutter. Địa chỉ này cần trỏ đúng đến IP hoặc tên miền nơi server Backend đang hoạt động để ứng dụng có thể gửi yêu cầu đến đúng nơi.

Khi cấu hình hoàn tất, ứng dụng có thể được chạy trực tiếp trên máy ảo hoặc thiết bị vật lý đã kết nối bằng lệnh flutter run. Để tạo gói cài đặt chính thức, sử dụng lệnh biên dịch như flutter build apk cho Android hoặc flutter build ios cho iOS, sau đó cài đặt tệp được tạo ra lên thiết bị.

5.3. Kiểm thử hệ thống

5.3.1. Kiểm thử module nhận diện khuôn mặt

Module nhận diện khuôn mặt dựa trên FaceNet được kiểm thử toàn diện để đảm bảo tính hiệu quả và chính xác. Quy trình kiểm thử bao gồm việc đánh giá khả năng phát hiện khuôn mặt trong nhiều điều kiện khác nhau. Hệ thống đã được thử nghiệm với các ảnh chụp từ nhiều góc độ, khoảng cách và điều kiện ánh sáng khác nhau nhằm mô phỏng môi trường thực tế mà ứng dụng sẽ hoạt động. Khả năng phát hiện nhiều khuôn mặt trong cùng một khung hình cũng được kiểm tra, đặc biệt quan trọng cho tính năng điểm danh hàng loạt do giáo viên thực hiện.

Việc đánh giá độ chính xác của các embedding vector là trọng tâm của quá trình kiểm thử. Chúng tôi đã so sánh các embedding được tạo ra từ nhiều ảnh khác nhau của cùng một người để xác định mức độ nhất quán. Vector embedding này phải đủ ổn định để nhận diện cùng một người trong các điều kiện chụp ảnh khác nhau, nhưng cũng phải đủ phân biệt để phân biệt giữa các cá nhân khác nhau. Độ tương đồng cosine

được sử dụng làm thước đo cho việc so sánh này, với ngưỡng giá trị 0.6 được chọn sau nhiều thử nghiệm để cân bằng giữa tỉ lệ nhận diện đúng và tỉ lệ báo động sai.

Quá trình xác thực khuôn mặt được kiểm thử nghiêm ngặt với cả trường hợp chuẩn và trường hợp ngoại lệ. Hệ thống phải có khả năng xác nhận học sinh đúng với độ chính xác cao, đồng thời từ chối những khuôn mặt không khớp với cơ sở dữ liệu. Các chỉ số quan trọng như FAR (False Acceptance Rate) và FRR (False Rejection Rate) đã được đo lường và phân tích để tối ưu hóa ngưỡng so sánh. Kết quả kiểm thử cho thấy mô hình đạt độ chính xác phát hiện khuôn mặt 98.2%, độ chính xác nhận diện 96.5%, với thời gian xử lý trung bình là 0.8 giây mỗi ảnh, đáp ứng đầy đủ yêu cầu về hiệu suất cho ứng dụng điểm danh trong thời gian thực.

5.3.2. Kiểm thử API Backend

API Backend của hệ thống đã được kiểm thử một cách hệ thống thông qua việc mô phỏng các tương tác thực tế từ client. Đầu tiên, các API xác thực học sinh được kiểm tra kỹ lưỡng để đảm bảo tính an toàn và độ tin cậy. Quá trình này bao gồm việc thử nghiệm đăng ký học sinh mới với các trường hợp hợp lệ và không hợp lệ, xác minh đăng nhập với thông tin chính xác và không chính xác, cũng như kiểm tra chức năng tham gia lớp học thông qua mã lớp. Các kịch bản tấn công phổ biến như SQL injection và brute force cũng được thử nghiệm để đảm bảo tính bảo mật của hệ thống.

API tải lên khuôn mặt được kiểm thử với nhiều loại dữ liệu đầu vào khác nhau. Việc xử lý các tình huống khi tải lên ảnh có khuôn mặt rõ ràng, ảnh không có khuôn mặt, hoặc ảnh có chất lượng kém cần được đánh giá cẩn thận. Quá trình chuyển đổi ảnh thành embedding và lưu trữ vào cơ sở dữ liệu cũng được theo dõi để đảm bảo tính nhất quán và hiệu quả. Các tình huống lỗi như mất kết nối giữa quá trình tải lên hoặc xử lý lỗi server cũng được mô phỏng để kiểm tra khả năng phục hồi của hệ thống.

Các API điểm danh, phần quan trọng nhất của hệ thống, được kiểm thử toàn diện trong điều kiện sử dụng thực tế. Các trường hợp điểm danh với khuôn mặt đúng, khuôn mặt sai, và điểm danh hàng loạt được thử nghiệm để đánh giá độ chính xác và hiệu suất của hệ thống. Khả năng xử lý các tình huống đặc biệt như học sinh đã điểm danh trước đó trong ngày hoặc điểm danh vào ngày khác cũng được kiểm tra kỹ lưỡng. Kết quả kiểm thử cho thấy API backend hoạt động ổn định và đáng tin cậy, với thời gian phản hồi phù hợp cho việc sử dụng trong môi trường thực tế.

5.3.3. Kiểm thử ứng dụng di động

Ứng dụng di động Flutter đã được kiểm thử trên nhiều thiết bị Android và iOS

khác nhau để đảm bảo tính tương thích và trải nghiệm người dùng nhất quán. Quá trình kiểm thử chức năng được thực hiện một cách có hệ thống, bắt đầu từ các tính năng cơ bản như đăng ký và đăng nhập, đến các chức năng chuyên biệt như tải lên ảnh khuôn mặt, tham gia lớp học, điểm danh qua nhận diện khuôn mặt và xem lịch sử điểm danh. Mỗi chức năng được thử nghiệm với các trường hợp đầu vào khác nhau để đảm bảo tính ổn định và khả năng xử lý lỗi của ứng dụng.

Giao diện người dùng được kiểm tra kỹ lưỡng trên các thiết bị với kích thước màn hình và độ phân giải khác nhau. Ứng dụng phải hiển thị chính xác, đáp ứng nhanh với các thao tác của người dùng và cung cấp phản hồi rõ ràng trong các tình huống lỗi hoặc thành công. Các yếu tố giao diện như bố cục, màu sắc, kích thước phông chữ và khả năng tiếp cận được đánh giá để đảm bảo trải nghiệm người dùng tốt nhất có thể. Thử nghiệm với người dùng thực cũng được tiến hành để thu thập phản hồi và cải thiện trải nghiệm người dùng.

Hiệu suất của ứng dụng là một khía cạnh quan trọng trong quá trình kiểm thử. Thời gian phản hồi của các thao tác khác nhau được đo lường để đảm bảo ứng dụng hoạt động mượt mà và nhanh chóng. Việc tiêu thụ tài nguyên như bộ nhớ, CPU và pin cũng được giám sát để tối ưu hóa hiệu suất. Các tình huống đặc biệt như mất kết nối mạng, chuyển đổi giữa nền trước và nền sau, hoặc khởi động lại ứng dụng cũng được mô phỏng để đảm bảo ứng dụng xử lý các trường hợp này một cách đúng đắn và không gây mất dữ liệu hay trải nghiệm người dùng kém. Kết quả kiểm thử cho thấy ứng dụng di động hoạt động ổn định, đáp ứng nhanh và cung cấp trải nghiệm người dùng thân thiện trên tất cả các thiết bị được thử nghiệm.

5.4. Kết chương

Các kết quả kiểm thử và đánh giá được trình bày trong chương này đã cung cấp những bằng chứng thực nghiệm vững chắc, khẳng định rằng hệ thống điểm danh được thiết kế và triển khai trong các chương trước hoạt động hiệu quả, ổn định và đáp ứng tốt các yêu cầu chức năng đã đề ra. Những kết quả này không chỉ xác nhận tính đúng đắn của các lựa chọn công nghệ và phương pháp tiếp cận mà còn là cơ sở quan trọng để đưa ra những kết luận tổng thể về thành công của đề tài và đề xuất các hướng phát triển, cải tiến trong tương lai

Chương 6. HƯỚNG PHÁT TRIỂN

6.1. Tổng kết kết quả đã đạt được

Nghiên cứu này đã thành công trong việc xây dựng một hệ thống điểm danh tự động dựa trên công nghệ nhận diện khuôn mặt hiện đại. Hệ thống đã tích hợp thành công mô hình FaceNet (20180402-114759) kết hợp với thuật toán phát hiện khuôn mặt MTCNN để đạt được độ chính xác nhận diện cao lên đến 96.5%. Kiến trúc back-end Flask cung cấp API đáng tin cậy và front-end Flutter mang đến trải nghiệm người dùng mượt mà trên các nền tảng di động.

Qua quá trình nghiên cứu và phát triển, hệ thống đã chứng minh nhiều ưu điểm so với phương pháp điểm danh truyền thống. Thứ nhất, phương pháp nhận diện khuôn mặt loại bỏ hoàn toàn việc gian lận điểm danh hộ, một vấn đề nan giải trong các hình thức điểm danh thông thường. Thứ hai, toàn bộ quy trình được tự động hóa, giảm thiểu thời gian và công sức cho cả giáo viên và học sinh. Thứ ba, hệ thống lưu trữ lịch sử điểm danh chi tiết với thời gian chính xác, tạo điều kiện thuận lợi cho việc theo dõi và báo cáo.

Tuy nhiên, trong quá trình phát triển và kiểm thử, một số hạn chế cũng đã được xác định, đặt nền tảng cho các cải tiến và hướng phát triển tương lai.

6.2. Cải thiện độ chính xác nhận diện khuôn mặt

Mặc dù mô hình FaceNet hiện tại đã đạt được độ chính xác nhận diện ấn tượng, vẫn còn dư địa để cải thiện, đặc biệt trong các điều kiện thách thức. Một số hướng phát triển để nâng cao độ chính xác nhận diện bao gồm:

6.2.1. Tăng cường dữ liệu đào tạo

Việc tăng số lượng ảnh cho mỗi học sinh với sự đa dạng về góc chụp, ánh sáng và biểu cảm sẽ giúp cải thiện đáng kể độ ổn định của vector embedding. Có thể triển khai kỹ thuật "data augmentation" để tạo ra nhiều biến thể từ các ảnh gốc như xoay, lật, điều chỉnh độ sáng, hoặc thêm nhiễu ngẫu nhiên. Đồng thời, có thể phát triển một thuật toán tự động yêu cầu học sinh chụp thêm ảnh khi hệ thống phát hiện độ tin cậy của embedding hiện tại không đủ cao.

6.2.2. Cải tiến phương pháp so sánh khuôn mặt

Phương thức hiện tại sử dụng độ tương đồng cosine với ngưỡng cố định (0.6) để nhận diện. Một cải tiến quan trọng là áp dụng ngưỡng động thay vì ngưỡng tĩnh, điều chỉnh dựa trên số lượng học sinh trong lớp và chất lượng embedding

6.3. Kết luận

Trong chương này, tôi đã trình bày các hướng phát triển tiềm năng cho hệ thống điểm danh dựa trên nhận diện khuôn mặt. Từ việc cải thiện độ chính xác của mô hình FaceNet thông qua tăng cường dữ liệu và phương pháp so sánh thích ứng, đến việc tăng cường an ninh bằng phát hiện sự sống và mã hóa dữ liệu sinh trắc học, những đề xuất này đều hướng tới một hệ thống hoàn thiện hơn. Phương pháp so sánh khuôn mặt dựa trên độ tương đồng cosine hiện tại đã chứng minh hiệu quả trong môi trường kiểm thử, nhưng vẫn có dư địa để cải tiến và tối ưu hóa cho các tình huống thực tế phức tạp hơn.

Các giải pháp tối ưu hóa hiệu suất và mở rộng quy mô sẽ đóng vai trò quan trọng khi hệ thống được triển khai rộng rãi. Việc áp dụng xử lý phân tán, tối ưu hóa trên thiết bị di động và thiết kế kiến trúc có khả năng mở rộng sẽ giúp hệ thống đáp ứng nhu cầu của các tổ chức giáo dục lớn. Đồng thời, việc tích hợp với các hệ thống quản lý học tập hiện có sẽ tạo ra một hệ sinh thái giáo dục toàn diện, thúc đẩy hiệu quả quản lý và nâng cao trải nghiệm học tập.

Qua việc không ngừng cải thiện trải nghiệm người dùng và áp dụng các nguyên tắc thiết kế hướng con người, hệ thống sẽ ngày càng trở nên thân thiện và dễ tiếp cận hơn với mọi đối tượng người dùng. Tóm lại, những hướng phát triển được đề xuất không chỉ giải quyết các hạn chế hiện tại của hệ thống mà còn mở ra tiềm năng ứng dụng rộng lớn hơn trong lĩnh vực giáo dục và công nghệ sinh trắc học, hướng tới một tương lai nơi việc điểm danh trở nên hoàn toàn tự động, chính xác và an toàn

TÀI LIỆU THAM KHẢO

[1] Bayer, M. Essential SQLAlchemy (2nd ed.). O'Reilly Media, 2020.

[2] Bradski, G. "The OpenCV Library". Dr. Dobb's Journal of Software Tools, 2000.

[3] Grinberg, M. Flask Web Development (2nd ed.). O'Reilly Media, 2018.Guzzi, V. Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games. Packt Publishing, 2021.

[4] Manning, C. D., Raghavan, P., & Schütze, H. Introduction to Information Retrieval. Cambridge University Press, 2008.

[5] Schroff, F., Kalenichenko, D., & Philbin, J. "FaceNet: A Unified Embedding for Face Recognition and Clustering". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823.

[6] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks". IEEE Signal Processing Letters, Vol. 23, No. 10, 2016, pp. 1499-1503.

[7] Adnan, W. A., Noor, N. L. M., "Material Design Guidelines: An Adoption Analysis in Mobile Applications", Journal of User Experience Design, Vol. 15, No. 2, 2022, pp. 78-93.