Tài liệu thiết kế

23020409 Đào Tự Phát
23020411 Cao Minh Quang
23020421 Hoàng Minh Quyền
23020437 Tạ Nguyên Thành

May 2025

Contents

1 Giới thiệu chung

1.1 Mục đích tài liệu

Mục đích của tài liệu này là cung cấp cái nhìn tổng quan và chi tiết về thiết kế hệ thống điểm danh nhân viên bằng nhận diện khuôn mặt. Tài liệu giúp định hướng phát triển, triển khai và bảo trì hệ thống, đồng thời làm cơ sở tham khảo cho các bên liên quan như đội phát triển, quản lý dự án và người dùng cuối.

1.2 Phạm vi hệ thống

Hệ thống điểm danh nhân viên được phát triển nhằm tự động hóa và tối ưu hóa quá trình quản lý chấm công trong doanh nghiệp. Hệ thống cho phép nhân viên thực hiện điểm danh nhanh chóng thông qua nhận diện khuôn mặt. Qua đó, giúp bộ phận nhân sự theo dõi thời gian làm việc một cách chính xác, minh bạch và tiết kiệm thời gian.
Hệ thống bao gồm hai phân quyền chính: User và Admin, mỗi vai trò có giao diện và chức năng riêng biệt, cụ thể như sau:

- User: Điểm danh bằng khuôn mặt và xem lịch sử điểm danh cá nhân.
- Admin: Quản lý người dùng, xem và lọc lịch sử điểm danh theo ngày hoặc theo nhân viên.

Hệ thống được thiết kế để triển khai trong môi trường nội bộ doanh nghiệp, hỗ trợ tối đa việc quản lý nhân sự, và có thể mở rộng trong tương lai nếu cần tích hợp thêm các tính năng khác như phân tích dữ liệu, cảnh báo hoặc tích hợp với hệ thống định vị GPS.

1.3 Đối tượng sử dụng tài liệu

- **Đội phát triển phần mềm**: để hiểu rõ yêu cầu và thiết kế hệ thống, phục vụ phát triển và triển khai.
- **Quản lý dự án**: để theo dõi tiến độ, quản lý chất lượng và tài nguyên dự án.
- **Người vận hành hệ thống**: để tham khảo khi triển khai và bảo trì.
- **Người dùng cuối (User/Admin)**: có thể tham khảo để hiểu rõ chức năng và quyền hạn của mình trong hệ thống.

1.4 Từ viết tắt và định nghĩa

- **User**: Người dùng hệ thống với quyền điểm danh và xem lịch sử điểm danh cá nhân.
- **Admin**: Người quản trị hệ thống, có quyền quản lý người dùng và xem toàn bộ lịch sử điểm danh.
- **DeepFace**: Thư viện AI hỗ trợ nhận diện khuôn mặt được tích hợp trong hệ thống.
- **JWT**: JSON Web Token, cơ chế xác thực bảo mật cho API.
- **ORM**: Object-Relational Mapping, kỹ thuật ánh xạ cơ sở dữ liệu sang đối tượng lập trình.

2 Yêu cầu chức năng

2.1 Yêu cầu chức năng

- **Đăng nhập hệ thống**
  - User và Admin đăng nhập bằng tài khoản và mật khẩu hợp lệ.
- **Điểm danh bằng nhận diện khuôn mặt (User)**
  - Chụp ảnh khuôn mặt qua webcam.
  - So sánh ảnh với dữ liệu khuôn mặt lưu trữ.
  - Ghi nhận thời gian điểm danh khi khuôn mặt được xác thực chính xác.
- **Xem lịch sử điểm danh (User)**
  - Hiển thị danh sách các lần điểm danh của chính người dùng theo ngày/tháng/năm.
- **Quản lý người dùng (Admin)**
  - Tạo mới, sửa đổi, xóa tài khoản người dùng.
  - Cập nhật thông tin cá nhân và ảnh khuôn mặt.
- **Xem và lọc lịch sử điểm danh (Admin)**
  - Tra cứu lịch sử điểm danh theo ngày.
  - Tra cứu lịch sử điểm danh của từng người dùng.
- **Phân quyền**
  - Phân biệt rõ quyền hạn giữa User và Admin.

2.2 Yêu cầu phi chức năng

- **Bảo mật**
  - Mã hóa dữ liệu đăng nhập và dữ liệu nhạy cảm.
  - Sử dụng JWT để xác thực và phân quyền API.
- **Hiệu năng**
  - Thời gian xử lý điểm danh dưới 3 giây kể từ khi chụp ảnh.
- **Tính sẵn sàng**
  - Hệ thống phải hoạt động 24/7, đảm bảo uptime trên 99%.
- **Khả năng mở rộng**

  - Hỗ trợ mở rộng số lượng người dùng và lịch sử điểm danh mà không giảm hiệu suất.

- **Tương thích**
  - Hỗ trợ các trình duyệt phổ biến hiện nay (Chrome, Firefox, Edge...).

2.3 Các ràng buộc hệ thống

- Phải sử dụng camera của thiết bị để chụp ảnh khuôn mặt tại thời điểm điểm danh.
- Ảnh khuôn mặt phải được lưu trữ và xử lý tuân thủ quy định bảo mật thông tin cá nhân.
- Hệ thống chạy trên môi trường nội bộ doanh nghiệp (mạng LAN hoặc VPN).
- Hạn chế truy cập trái phép qua các cơ chế bảo mật và phân quyền.
- Giới hạn dung lượng lưu trữ ảnh và dữ liệu điểm danh phù hợp với quy mô doanh nghiệp.

3 Thiết kế kiến trúc

3.1 Kiến trúc tổng thể

Trong dự án hệ thống điểm danh lần này, nhóm chúng em áp dụng kiểu kiến trúc **Client - Server** để xây dựng hệ thống. Cụ thể, hệ thống được chia thành hai phần riêng biệt:

- **Client**: Giao diện web được phát triển bằng React, chạy trên trình duyệt web (web browser), đóng vai trò là phía người dùng sử dụng. Client chịu trách nhiệm thu thập dữ liệu từ người dùng như hình ảnh khuôn mặt để điểm danh, truy vấn lịch sử điểm danh,... Sau đó gửi yêu cầu đến phía Server thông qua các API call.
- **Server**: Phần backend sử dụng FastAPI, kết hợp với Tortoise ORM và cơ sở dữ liệu PostgreSQL để xử lý logic nghiệp vụ, quản lý người dùng và bản ghi điểm danh. Khi nhận được yêu cầu từ client, server sẽ xử lý dữ liệu, gọi thư viện nhận diện khuôn mặt (DeepFace), đối chiếu với dữ liệu đã lưu, cập nhật kết quả và trả phản hồi lại cho client.

Ngoài ra, hệ thống còn tích hợp **Async/Await** để xử lý bất đồng bộ, giúp hệ thống phản hồi nhanh và hỗ trợ nhiều người dùng truy cập cùng lúc mà không bị nghẽn hoặc treo server. Tính năng này đặc biệt quan trọng trong các khung giờ cao điểm như đầu giờ sáng khi nhiều nhân viên đồng loạt điểm danh.
Cơ sở dữ liệu được thiết kế để lưu trữ thông tin User(username, password, fullname, email, ..), bản ghi điểm danh Attendance(user, date, time, status, ...),

hỗ trợ thao tác CRUD dễ dàng thông qua ORM hoặc truy vấn SQL trực tiếp. Với cách thiết kế này, hệ thống điểm danh hoạt động hiệu quả, nhận diện khuôn mặt chính xác, API phản hồi nhanh chóng, đảm bảo trải nghiệm mượt mà và tiện lợi cho người dùng cuối.
Có thể mở rộng thành kiến trúc **Microservices** trong tương lai nếu cần phân tách các module riêng biệt như dịch vụ nhận diện khuôn mặt, quản lý người dùng, lưu trữ dữ liệu điểm danh.

3.2 Các thành phần chính

Frontend

- Được phát triển bằng React – một thư viện JavaScript mạnh mẽ cho việc xây dựng giao diện người dùng.
- Cung cấp hai giao diện chính:
  - **Giao diện người dùng (User)**: đăng nhập, điểm danh bằng khuôn mặt, xem lịch sử điểm danh cá nhân.
  - **Giao diện quản trị (Admin)**: quản lý người dùng, xem và lọc lịch sử điểm danh theo ngày hoặc theo người dùng.

Backend

- Sử dụng FastAPI – một framework Python hiện đại và hiệu suất cao chuyên dùng xây dựng API.
- Đảm nhiệm các chức năng cốt lõi:
  - Xác thực và phân quyền người dùng (User/Admin)
  - Xử lý điểm danh bằng khuôn mặt
  - Truy vấn và tổng hợp lịch sử điểm danh
  - Xử lý API bảo mật với JWT và các biện pháp xác thực khác
- Tích hợp thư viện DeepFace để xử lý các tác vụ liên quan đến khuôn mặt.
- Đóng vai trò trung gian giữa frontend và cơ sở dữ liệu.

3.3 Cơ sở dữ liệu

- Sử dụng **PostgreSQL (Neon)** – dịch vụ PostgreSQL trên nền tảng cloud.
- Áp dụng **Tortoise ORM** để ánh xạ các bảng dữ liệu sang mô hình Python, giúp quản lý và truy vấn dễ dàng.
- Lưu trữ thông tin quan trọng:
  - User (username, password, fullname, email, ...)
  - Attendance(user, date, time, status, ...)

3.4 Luồng hoạt động tổng quan

1. Người dùng (Quản lý) tương tác trên **Frontend** (đăng nhập, điểm danh, xem lịch sử, ...).
2. Frontend gửi yêu cầu HTTP API tới **Backend** kèm dữ liệu (ảnh khuôn mặt, thông tin truy vấn, ...).
3. Backend nhận dữ liệu, xử lý nghiệp vụ:
   - Xác thực người dùng, phân quyền.
   - Gửi ảnh khuôn mặt tới **DeepFace** để nhận diện.
   - Lưu hoặc truy vấn dữ liệu điểm danh trong **cơ sở dữ liệu**.
4. Backend trả kết quả về Frontend.
5. Frontend hiển thị kết quả cho người dùng.

3.5 Biểu đồ Use Case

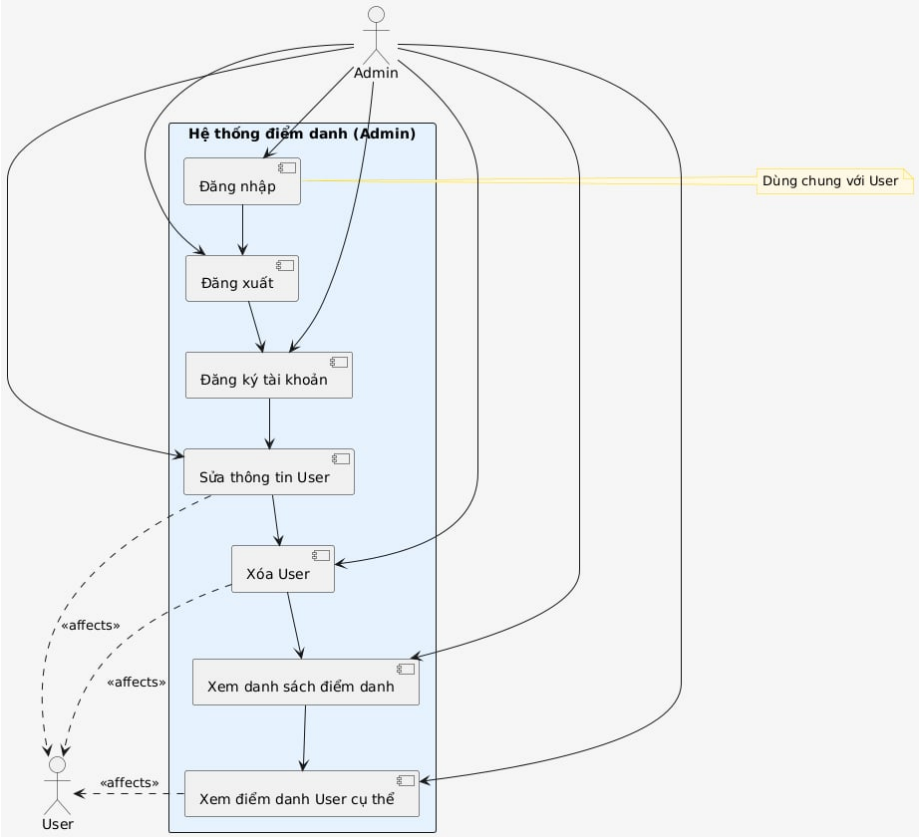

Figure 1: Biểu đồ Use Case cho người quản lý (Admin)

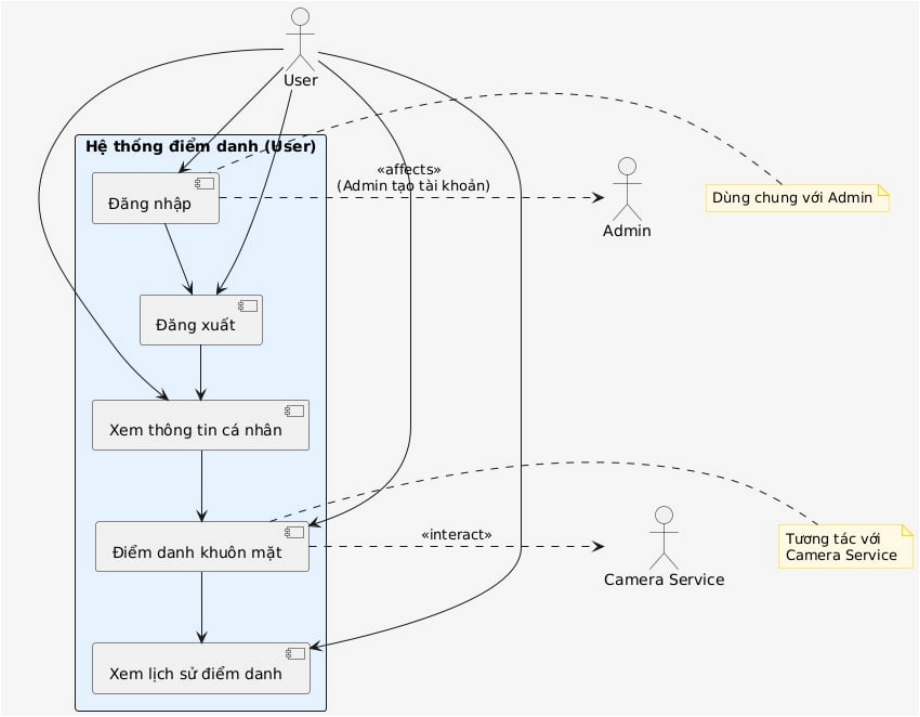

Figure 2: Biểu đồ Use Case cho người dùng (User)

4 Thiết kế cơ sở dữ liệu

Hệ thống sử dụng cơ sở dữ liệu **PostgreSQL (Neon)** - dịch vụ PostgreSQL trên nền tảng cloud kết hợp với **Tortoise ORM** để quản lý và tương tác dữ liệu. Cấu trúc cơ sở dữ liệu được thiết kế theo mô hình thực thể quan hệ.

4.1 Thực thể

4.1.1 Bảng `users`

Lưu trữ thông tin tài khoản và hồ sơ của người dùng.

Tên cột	Kiểu dữ liệu	Ràng buộc	Mô tả
`user_id`	INT	PK, AUTO INCREMENT	ID người dùng (duy nhất)
`username`	VARCHAR(50)	NOT NULL, UNIQUE	Tên đăng nhập
`password_hash`	VARCHAR(255)	NOT NULL	Mật khẩu đã mã hóa
`role`	VARCHAR(20)	DEFAULT 'user'	Vai trò: 'user' hoặc 'admin'
`fullname`	VARCHAR(255)	NOT NULL	Họ và tên đầy đủ
`email`	VARCHAR(255)	NOT NULL, UNIQUE	Địa chỉ email
`department`	VARCHAR(255)	NOT NULL	Khoa/Bộ môn công tác
`face_path`	VARCHAR(255)	NOT NULL, UNIQUE	Đường dẫn ảnh khuôn mặt

Table 1: Cấu trúc bảng `users`

4.1.2 Bảng `attendances`

Lưu trữ lịch sử điểm danh của người dùng.

Tên cột	Kiểu dữ liệu	Ràng buộc	Mô tả
`attendance_id`	INT	PK, AUTO INCREMENT	ID điểm danh
`user_id`	INT	FK, ON DELETE CASCADE	Liên kết với người dùng
`date`	DATE	NOT NULL	Ngày điểm danh
`time`	TIME	NULLABLE	Thời gian điểm danh
`status`	VARCHAR(20)	DEFAULT 'pending'	Trạng thái điểm danh

Table 2: Cấu trúc bảng `attendances`

4.2 Quan hệ

- Một người dùng (`users`) có thể có nhiều bản ghi điểm danh (`attendances`).
- Mỗi bản ghi điểm danh liên kết với một người dùng duy nhất thông qua khóa ngoại `user_id`.

4.3 Lợi ích của thiết kế

Thiết kế này đảm bảo:

- Dễ dàng truy vấn thông tin điểm danh theo người dùng hoặc theo ngày.
- Đảm bảo tính toàn vẹn dữ liệu nhờ ràng buộc khoá ngoại và chuẩn hóa bảng.

5 Thiết kế giao diện

5.1 Chi tiết thiết kế

5.1.1 Trang chủ (Đăng nhập)

Name: Trang đăng nhập vào hệ thống
Description: Giao diện đăng nhập cho phép người dùng truy cập vào hệ thống điểm danh khuôn mặt
Route: `/`
File: `app/page.tsx`
Components:

- **LoginForm:**
  - Trường nhập tên đăng nhập (username input field)
  - Trường nhập mật khẩu (password input field)
  - Nút "Đăng nhập" (login button)
  - Hiển thị thông báo lỗi khi thông tin không hợp lệ

Layout:

- Tiêu đề: "Hệ thống điểm danh"
- Mô tả: "Đăng nhập để tiếp tục"
- Thiết kế: Form đăng nhập được căn giữa màn hình, nền trắng bo góc, có bóng đổ, nền tổng thể màu xám nhạt

5.1.2 Trang người dùng (User Dashboard)

Name: Dashboard dành cho người dùng
Description: Giao diện chính cho người dùng sau khi đăng nhập thành công
Route: `/user`
File: `app/user/page.tsx`
Components:

- **UserDashboard:**
  - **UserProfile:** Hiển thị thông tin cá nhân (họ tên, email, phòng ban, ảnh khuôn mặt)
  - **AttendanceHistory:** Bảng lịch sử điểm danh với các cột: ngày, giờ, trạng thái
  - **AttendanceButton:** Nút cho phép người dùng thực hiện điểm danh (nếu có)
  - **StatusNotification:** Hiển thị thông báo trạng thái thành công/thất bại

Layout:

- Dashboard được chia thành các section rõ ràng
- Thông tin cá nhân hiển thị ở phần trên
- Lịch sử điểm danh và các chức năng khác hiển thị ở phần dưới

5.1.3 Trang quản trị (Admin Dashboard)

Name: Dashboard dành cho quản trị viên
Description: Giao diện quản lý toàn bộ hệ thống dành cho admin
Route: `/admin`
File: `app/admin/page.tsx`
Components:

- **AdminToolbar:**
  - Nút "Xem điểm danh hôm nay" (chuyển đến `/admin/attendance`)
  - Nút "Đăng xuất"
- **TabNavigation:**
  - **Tab 1: Danh sách người dùng**
    - **UserList:** Bảng danh sách với các cột: họ tên, email, phòng ban, vai trò, ảnh khuôn mặt, thao tác
    - **FilterOptions:** Bộ lọc theo phòng ban, vai trò
    - **SearchBox:** Tìm kiếm theo tên hoặc email
    - **ActionButtons:** Các nút thao tác: sửa, xóa, xem lịch sử, đổi vai trò
    - **EditUserDialog:** Form chỉnh sửa thông tin người dùng
    - **DeleteConfirmDialog:** Dialog xác nhận xóa người dùng
    - **RoleChangeDialog:** Dialog xác nhận đổi vai trò
  - **Tab 2: Thêm người dùng mới**
    - **AddUserForm:** Form tạo người dùng mới với các trường: tên đăng nhập, họ tên, email, mật khẩu, phòng ban, vai trò
    - **FaceImageUpload:** Chức năng upload ảnh khuôn mặt (chọn file hoặc chụp từ camera)
    - **ImagePreview:** Xem trước ảnh đã chọn
    - **SubmitButton:** Nút "Thêm người dùng"
    - **StatusMessage:** Thông báo kết quả thành công/lỗi

Layout:

- Thanh công cụ hiển thị ở phần trên
- Tab navigation cho phép chuyển đổi giữa các chức năng chính
- Nội dung của từng tab hiển thị ở phần chính

5.1.4 Trang điểm danh hôm nay (Admin - Daily Attendance)

Name: Danh sách điểm danh hàng ngày
Description: Hiển thị thông tin điểm danh của tất cả nhân viên trong ngày hiện tại
Route: `/admin/attendance`
File: `app/admin/attendance/page.tsx`
Components:

- **DailyAttendanceList:**
  - Bảng với các cột: họ tên, phòng ban, thời gian điểm danh, trạng thái, ghi chú
  - **DepartmentFilter:** Bộ lọc theo phòng ban
  - **NameSearch:** Tìm kiếm theo tên nhân viên

Layout:

- Tiêu đề: "Điểm danh hôm nay"
- Bảng danh sách chiếm phần chính của trang
- Các bộ lọc và tìm kiếm hiển thị ở phần trên bảng

5.1.5 Trang lịch sử điểm danh từng người dùng (Admin - User Attendance History)

Name: Lịch sử điểm danh chi tiết của một nhân viên
Description: Hiển thị toàn bộ lịch sử điểm danh của một người dùng cụ thể
Route: `/admin/user-attendance/[userId]?name=...`
File: `app/admin/user-attendance/[userId]/page.tsx`
Components:

- **UserAttendanceHistory:**
  - Bảng với các cột: ngày, giờ, trạng thái, ghi chú
  - **DateFilter:** Bộ lọc theo ngày/tháng
- **LoadingState:** Hiệu ứng loading khi dữ liệu chưa sẵn sàng

Layout:

- Tiêu đề: "Lịch sử điểm danh: [Tên người dùng]"
- Bảng lịch sử chiếm phần chính
- Bộ lọc ngày tháng hiển thị ở phần trên

5.1.6 Các giao diện phụ trợ và thành phần dùng chung

Name: Shared Components
Description: Các thành phần giao diện được sử dụng chung trong toàn bộ hệ thống
Components:

- **AlertDialog:** Dialog xác nhận cho các thao tác quan trọng (xóa, đổi vai trò)
- **ToasterNotification:** Hiển thị thông báo thành công/thất bại
- **TabNavigation:** Thanh điều hướng tab cho admin dashboard
- **FormInputs:** Các component form tái sử dụng cho thêm/sửa người dùng
- **LoadingSpinner:** Hiệu ứng loading cho các trang có thời gian tải dữ liệu

Design Guidelines:

- Sử dụng design system thống nhất
- Màu sắc và typography nhất quán
- Responsive design cho mọi thiết bị

5.2 Ghi chú tổng quan về thiết kế

Design Philosophy:

- **Phong cách:** Hiện đại, tối giản với màu nền trung tính, khung bo góc, bóng đổ nhẹ
- **Tương thích:** Responsive design hoạt động tốt trên cả desktop và mobile
- **Trải nghiệm người dùng:**
  - Thông báo rõ ràng và dễ hiểu
  - Thao tác nhanh chóng và trực quan
  - Xác nhận các hành động quan trọng để tránh thao tác nhầm
  - Admin dashboard tập trung tất cả chức năng quản lý trong một trang với điều hướng tab
- **Accessibility:** Tuân thủ các nguyên tắc thiết kế accessible để đảm bảo khả năng tiếp cận

6 Thiết kế tính năng

6.1 Tính năng của người quản lý (Admin)

6.1.1 Đăng ký tài khoản

Name: Đăng ký tài khoản mới
Actor: Admin
Goal: Admin tạo tài khoản mới thành công để có thể sử dụng với vai trò *Admin* hoặc *User*
Pre-condition:

- Tài khoản trước đó chưa tồn tại
- Hệ thống hiển thị giao diện đăng ký tài khoản mới

Main Flow:

1. Admin nhập thông tin cần thiết của mình (*Họ tên, tên đăng nhập, mật khẩu, email, phòng ban, vai trò Admin hoặc User*) và nhấn nút xác nhận đăng ký
2. Hệ thống kiểm tra tính hợp lệ của thông tin
3. Hệ thống kiểm tra thông tin đã tồn tại trong *database* hay chưa
4. Hệ thống chụp ảnh mặt của Admin/User và lưu vào kho ảnh
5. Hệ thống mã hoá mật khẩu và lưu thông tin của Admin/User vào *database*
6. Hệ thống thông báo đã tạo tài khoản thành công cho Admin/User

Alternative Flow:

- **Bước 1:** Nếu Admin huỷ bỏ đăng ký tài khoản hoặc chuyển hướng sang trang khác trước khi ấn nút xác nhận đăng ký, hệ thống huỷ bỏ yêu cầu và không lưu thông tin đã nhập
- **Bước 2a:** Nếu Admin nhập không đầy đủ phần thông tin, hệ thống báo lỗi và yêu cầu nhập lại
- **Bước 2b:** Nếu thông tin không hợp lệ theo format đã đặt ra, hệ thống báo lỗi và yêu cầu nhập lại
- **Bước 3:** Nếu thông tin Admin đã tồn tại trên database, hệ thống báo lỗi và yêu cầu nhập lại
- **Bước 4:** Nếu chụp ảnh lỗi, hệ thống yêu cầu chụp lại ảnh khác
- **Bước 5:** Nếu lỗi khi lưu thông tin, hệ thống báo lỗi kỹ thuật và yêu cầu thử lại sau

Post-condition:

- Admin tạo được tài khoản thành công và có thể sử dụng để đăng nhập
- Hệ thống lưu lại thông tin Admin trên cơ sở dữ liệu
- Tài khoản có thể thuộc vai trò Admin hoặc User

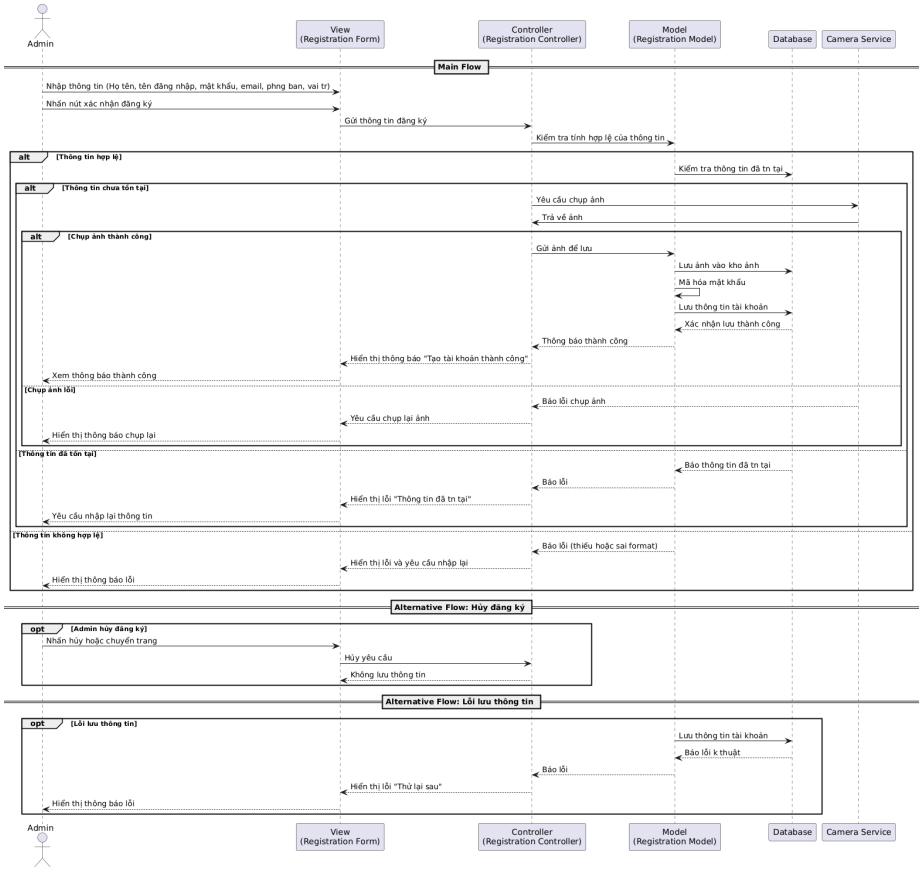

Figure 3: Biểu đồ tuần tự cho ca sử dụng đăng kí tài khoản mới

6.1.2 Đăng nhập tài khoản

Name: Đăng nhập vào tài khoản
Actor: Admin
Goal: Admin đăng nhập được vào tài khoản đã đăng ký từ trước để quản lý hệ thống
Pre-condition:

- Tài khoản đã được đăng ký hợp lệ từ trước

- Hệ thống hiển thị giao diện đăng nhập

Main Flow:

1. Admin nhập thông tin đăng nhập gồm *tên đăng nhập, mật khẩu* và nhấn nút xác nhận đăng nhập
2. Hệ thống kiểm tra tính hợp lệ của định dạng thông tin đầu vào
3. Hệ thống kiểm tra thông tin đăng nhập có tồn tại và khớp trong *database* hay không
4. Hệ thống thông báo đăng nhập thành công và chuyển hướng Admin đến giao diện quản lý hệ thống

Alternative Flow:

- **Bước 1:** Nếu Admin huỷ bỏ thao tác đăng nhập hoặc chuyển hướng sang trang khác trước khi nhấn nút xác nhận, hệ thống huỷ bỏ yêu cầu và không xử lý thông tin đã nhập
- **Bước 2:** Nếu định dạng thông tin không hợp lệ (ví dụ: tên đăng nhập chứa ký tự đặc biệt không cho phép), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại
- **Bước 3a:** Nếu tên đăng nhập không tồn tại hoặc mật khẩu không đúng, hệ thống báo lỗi "Sai thông tin đăng nhập" và yêu cầu nhập lại
- **Bước 3b:** Nếu có lỗi hệ thống hoặc lỗi kết nối đến *database*, hệ thống hiển thị thông báo lỗi kỹ thuật và yêu cầu thử lại sau

Post-condition:

- Admin đăng nhập thành công và được chuyển đến giao diện quản lý
- Hệ thống ghi nhận trạng thái đăng nhập (phiên làm việc) của Admin

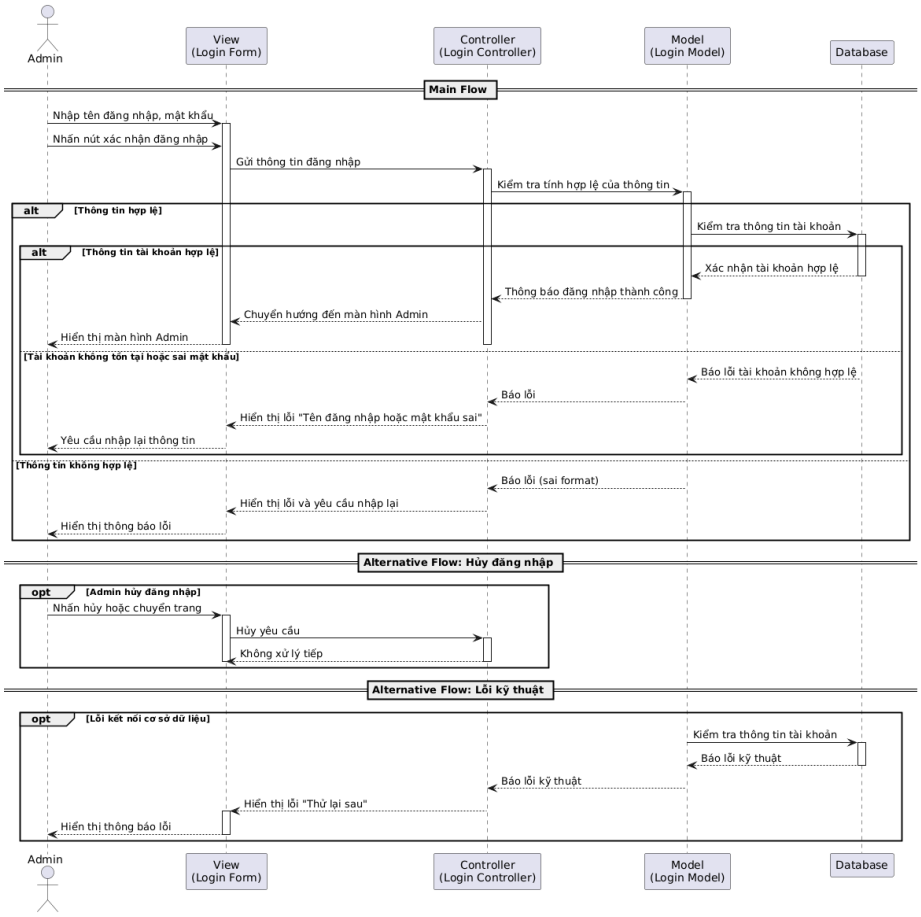

Figure 4: Biểu đồ tuần tự cho ca sử dụng đăng nhập tài khoản người quản lý

6.1.3 Đăng xuất tài khoản

Name: Đăng xuất
Actor: Admin
Goal: Admin đăng xuất ra khỏi hệ thống
Pre-condition:

- Tài khoản của Admin đã đăng nhập vào hệ thống
- Hệ thống hiển thị giao diện có chức năng đăng xuất

Main Flow:

1. Admin nhấn vào nút "Đăng xuất"
2. Hệ thống xử lý yêu cầu và đăng xuất tài khoản Admin

3. Hệ thống hiển thị thông báo đăng xuất thành công và chuyển hướng đến giao diện đăng nhập

Alternative Flow:

- **Bước 2:** Nếu xảy ra lỗi trong quá trình đăng xuất (ví dụ: lỗi hệ thống hoặc không xoá được phiên làm việc), hệ thống hiển thị thông báo lỗi và yêu cầu thử lại

Post-condition:

- Tài khoản Admin đã được đăng xuất khỏi hệ thống
- Hệ thống quay về giao diện đăng nhập, sẵn sàng cho phiên làm việc mới

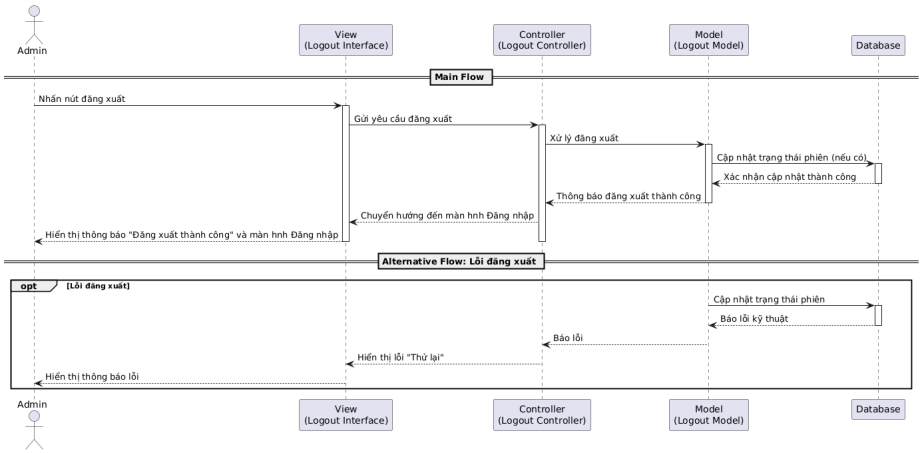

Figure 5: Biểu đồ tuần tự cho ca sử dụng đăng xuất tài khoản người quản lý

6.1.4 Sửa thông tin của User

Name: Sửa thông tin của User
Actor: Admin
Goal: Admin chỉnh sửa và cập nhật thành công thông tin cá nhân của User
Pre-condition:

- Admin đã đăng nhập vào hệ thống
- User là một tài khoản hợp lệ và tồn tại trong *database*
- Hệ thống hiển thị giao diện chỉnh sửa thông tin User

Main Flow:

1. Admin thay đổi các thông tin cần chỉnh sửa của User (ví dụ: *họ tên, email, phòng ban, vai trò, ảnh đại diện,...*) và nhấn nút xác nhận chỉnh sửa

2. Hệ thống kiểm tra tính hợp lệ của các thông tin đã nhập
3. Hệ thống xác minh rằng tài khoản User cần chỉnh sửa có tồn tại trong *database*
4. Hệ thống lưu thông tin mới vào *database*
5. Hệ thống hiển thị thông báo cập nhật thông tin thành công

Alternative Flow:

- **Bước 1:** Nếu Admin huỷ thao tác chỉnh sửa hoặc chuyển sang trang khác trước khi xác nhận, hệ thống huỷ bỏ yêu cầu và không lưu thông tin
- **Bước 2a:** Nếu Admin nhập thiếu thông tin bắt buộc, hệ thống hiển thị thông báo lỗi và yêu cầu bổ sung
- **Bước 2b:** Nếu thông tin nhập không đúng định dạng quy định (ví dụ: email không hợp lệ, số điện thoại sai cấu trúc), hệ thống yêu cầu nhập lại
- **Bước 2c:** Nếu có lỗi khi tải ảnh đại diện mới, hệ thống hiển thị thông báo lỗi và yêu cầu thử lại
- **Bước 3:** Nếu tài khoản User không tồn tại trong *database*, hệ thống thông báo lỗi và không thực hiện chỉnh sửa
- **Bước 4:** Nếu hệ thống gặp lỗi trong quá trình lưu thông tin (ví dụ: lỗi kết nối cơ sở dữ liệu), hệ thống hiển thị thông báo lỗi kỹ thuật và yêu cầu thử lại sau

Post-condition:

- Thông tin mới của User được cập nhật thành công vào cơ sở dữ liệu
- Hệ thống phản ánh thông tin User mới trên giao diện tương ứng

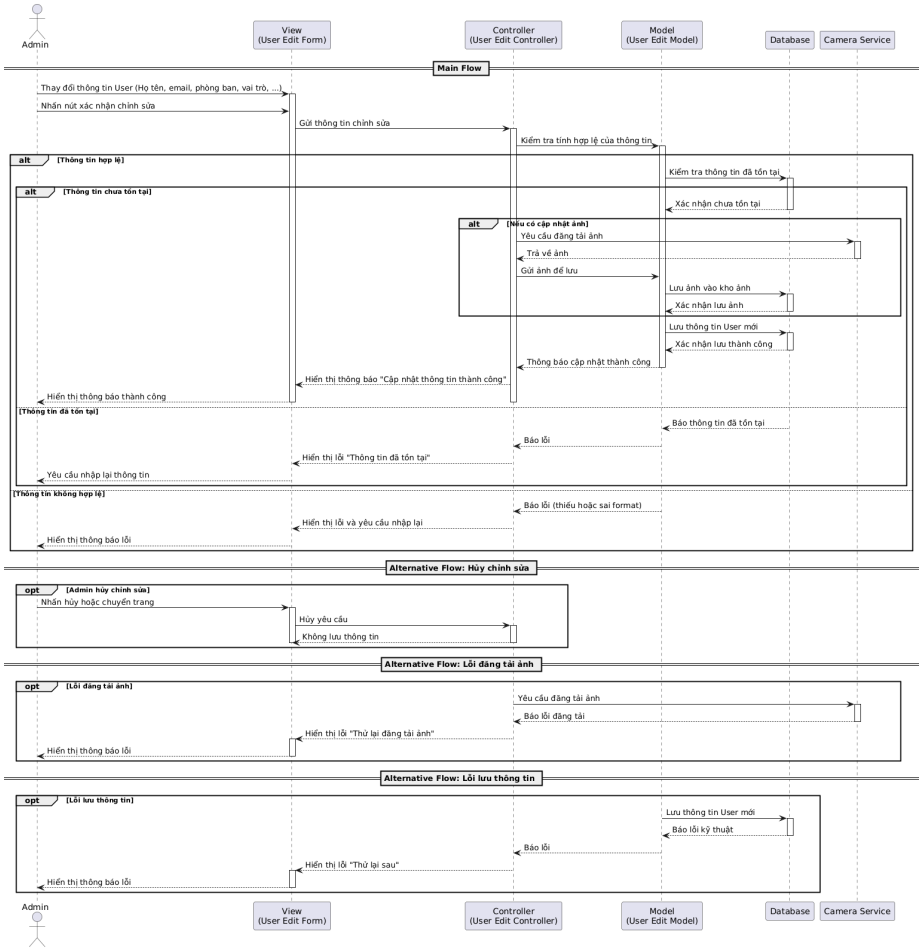

Figure 6: Biểu đồ tuần tự cho ca sử dụng sửa thông tin user

6.1.5 Xoá User khỏi hệ thống

Name: Xoá đi một User
Actor: Admin
Goal: User được Admin chọn sẽ bị xoá khỏi hệ thống
Pre-condition:

- Admin đã đăng nhập vào hệ thống
- User cần xoá là một tài khoản hợp lệ và đang tồn tại trong *database*
- Hệ thống hiển thị giao diện danh sách User cho phép thao tác xoá

Main Flow:

1. Admin chọn User cần xoá và nhấn vào nút “Xoá”
2. Hệ thống xác minh tài khoản và thực hiện xoá User cùng toàn bộ thông tin liên quan khỏi *database*
3. Hệ thống hiển thị thông báo xoá thành công và quay về màn hình chính của Admin

Alternative Flow:

- **Bước 1:** Nếu User được chọn đã bị xoá hoặc không tồn tại trong hệ thống tại thời điểm thao tác, hệ thống hiển thị thông báo lỗi “Tài khoản không tồn tại” và yêu cầu thử lại
- **Bước 2:** Nếu xảy ra lỗi khi xoá thông tin trong *database* (ví dụ: lỗi kết nối hoặc lỗi ràng buộc dữ liệu), hệ thống hiển thị lỗi kỹ thuật và yêu cầu thử lại sau

Post-condition:

- User được xoá hoàn toàn khỏi hệ thống và không còn tồn tại trong *database*
- Giao diện hệ thống cập nhật lại danh sách User hiện hành

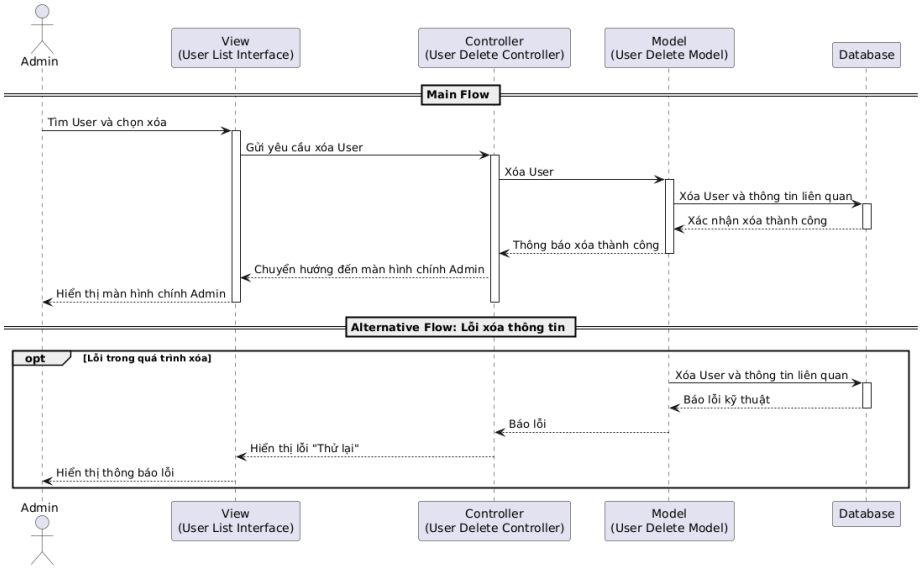

Figure 7: Biểu đồ tuần tự cho ca sử dụng xoá user khỏi hệ thống

6.1.6 Xem danh sách điểm danh hàng ngày

Name: Xem danh sách điểm danh hàng ngày
Actor: Admin

Goal: Hệ thống trả về danh sách điểm danh hàng ngày cho Admin
Pre-condition:

- Admin đã đăng nhập vào hệ thống
- Hệ thống hiển thị giao diện màn hình chính của Admin

Main Flow:

1. Admin nhấn vào nút "Xem danh sách điểm danh hàng ngày"
2. Hệ thống truy vấn cơ sở dữ liệu và hiển thị danh sách điểm danh của tất cả User trong ngày hiện tại (bao gồm: *tên User, thời gian điểm danh, trạng thái điểm danh,...*)

Alternative Flow:

- **Bước 1:** Nếu có lỗi trong quá trình hiển thị (ví dụ: lỗi giao diện, lỗi trình duyệt), hệ thống báo lỗi và yêu cầu thử lại
- **Bước 2:** Nếu hệ thống gặp lỗi kết nối cơ sở dữ liệu khi truy vấn thông tin điểm danh, hệ thống báo lỗi kỹ thuật và yêu cầu thử lại sau

Post-condition:

- Hệ thống hiển thị danh sách điểm danh trong ngày thành công để Admin theo dõi và quản lý

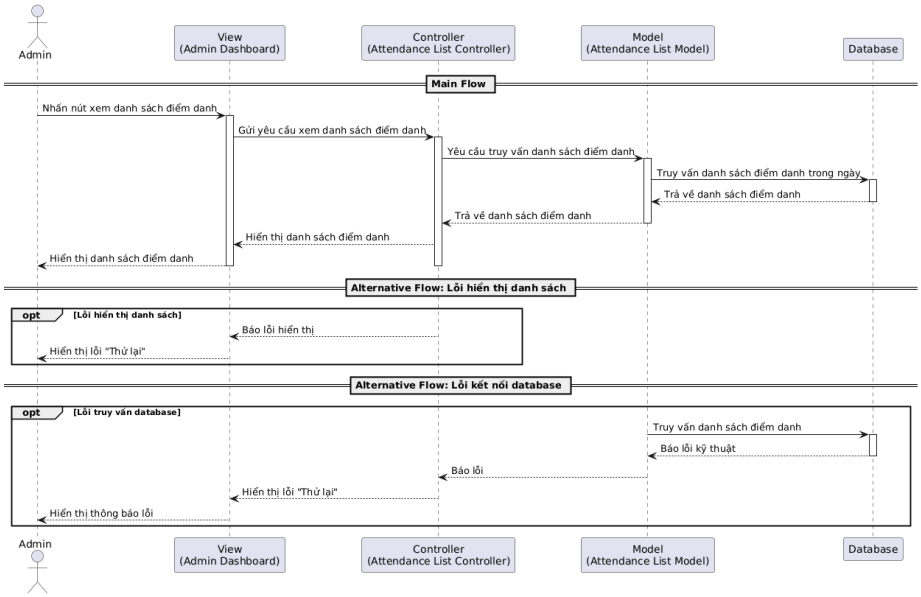

Figure 8: Biểu đồ tuần tự cho ca sử dụng xem danh sách điểm danh hàng ngày

6.1.7 Xem thông tin điểm danh của một User cụ thể

Name: Xem thông tin điểm danh của một User cụ thể
Actor: Admin
Goal: Hệ thống trả về thông tin điểm danh của một User cụ thể cho Admin
Pre-condition:

- Admin đã đăng nhập vào hệ thống
- User được chọn là tài khoản tồn tại hợp lệ trong *database*
- Hệ thống hiển thị giao diện danh sách điểm danh tổng hợp của các User

Main Flow:

1. Admin chọn một User cụ thể từ danh sách điểm danh
2. Hệ thống truy vấn và hiển thị thông tin điểm danh của User, bao gồm: *mã điểm danh, ngày tháng năm, thời gian, trạng thái (đã điểm danh, chưa điểm danh,...)*

Alternative Flow:

- **Bước 1:** Nếu có lỗi khi truy vấn thông tin User (ví dụ: lỗi hệ thống, lỗi dữ liệu), hệ thống báo lỗi và yêu cầu thử lại
- **Bước 2:** Nếu User chưa có thông tin điểm danh nào, hệ thống hiển thị kết quả rỗng và thông báo "Chưa có dữ liệu điểm danh"

Post-condition:

- Hệ thống hiển thị đầy đủ thông tin điểm danh của User để Admin kiểm tra và xử lý nếu cần thiết

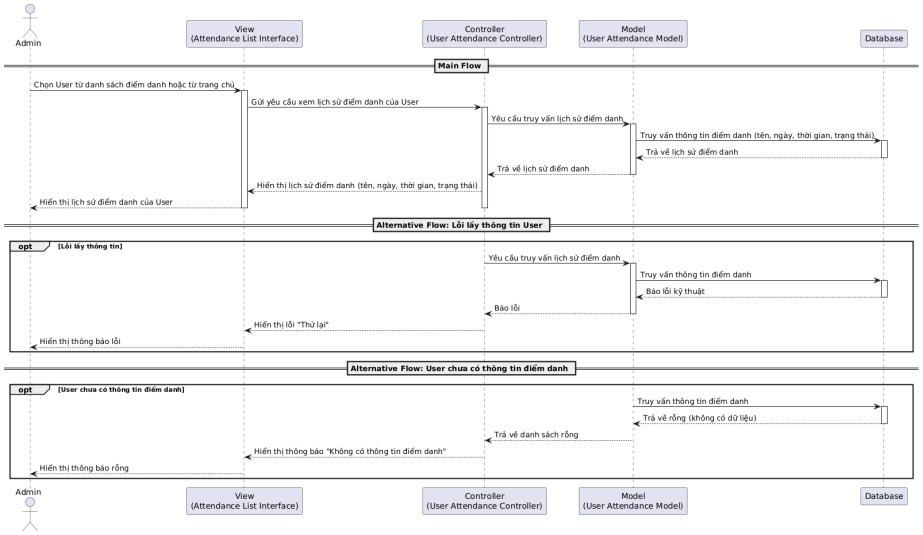

Figure 9: Biểu đồ tuần tự cho ca sử dụng Xem thông tin điểm danh của một người dùng cụ thể

6.2 Tính năng của người dùng (User)

6.2.1 Đăng nhập tài khoản của User

Name: Đăng nhập vào tài khoản
Actor: User
Goal: User đăng nhập vào tài khoản đã được Admin đăng ký để sử dụng hệ thống
Pre-condition:

- Tài khoản User đã được Admin tạo và lưu trong hệ thống
- Hệ thống hiển thị giao diện đăng nhập

Main Flow:

1. User nhập *tên đăng nhập, mật khẩu* và nhấn nút xác nhận đăng nhập
2. Hệ thống kiểm tra tính hợp lệ của thông tin đầu vào
3. Hệ thống kiểm tra thông tin đăng nhập có khớp trong *database* hay không
4. Hệ thống hiển thị thông báo đăng nhập thành công và chuyển hướng User đến giao diện người dùng của mình

Alternative Flow:

- **Bước 1:** Nếu User huỷ bỏ thao tác đăng nhập hoặc chuyển hướng sang trang khác trước khi xác nhận, hệ thống huỷ bỏ yêu cầu và không xử lý thông tin

- **Bước 2:** Nếu định dạng tên đăng nhập hoặc mật khẩu không hợp lệ (ví dụ: bỏ trống, chứa ký tự cấm), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại
- **Bước 3a:** Nếu tên đăng nhập không tồn tại hoặc mật khẩu không chính xác, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại
- **Bước 3b:** Nếu xảy ra lỗi kỹ thuật (ví dụ: lỗi kết nối cơ sở dữ liệu), hệ thống thông báo lỗi và yêu cầu thử lại sau

Post-condition:

- User đăng nhập thành công và được chuyển đến giao diện làm việc của mình
- Hệ thống khởi tạo phiên làm việc cho User

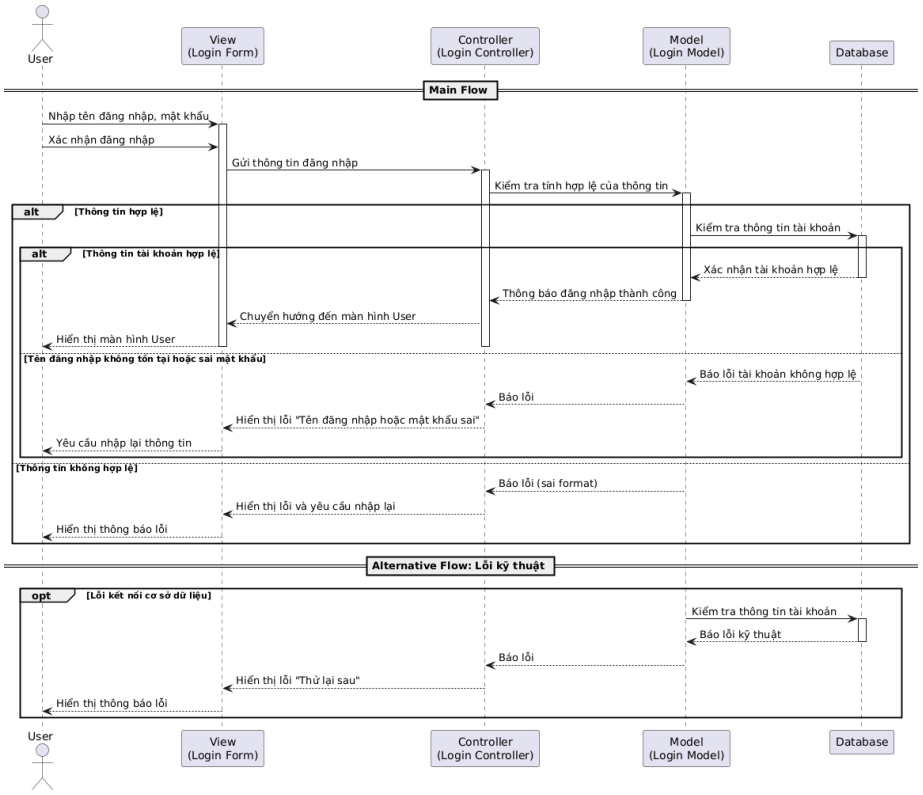

Figure 10: Biểu đồ tuần tự cho ca sử dụng đăng nhập tài khoản người dùng

6.2.2 Đăng xuất tài khoản của User

Name: Đăng xuất
Actor: User
Goal: User đăng xuất khỏi hệ thống một cách an toàn và kết thúc phiên làm việc
Pre-condition:

- Tài khoản của User đã đăng nhập vào hệ thống
- Giao diện hiện tại có hiển thị tuỳ chọn đăng xuất

Main Flow:

1. User nhấn vào nút "Đăng xuất"
2. Hệ thống xử lý yêu cầu và đăng xuất tài khoản User khỏi phiên làm việc
3. Hệ thống hiển thị thông báo đăng xuất thành công và chuyển hướng đến giao diện đăng nhập

Alternative Flow:

- **Bước 2:** Nếu trong quá trình đăng xuất xảy ra lỗi (ví dụ: lỗi session hoặc lỗi hệ thống), hệ thống hiển thị thông báo lỗi kỹ thuật và yêu cầu thử lại

Post-condition:

- User đã được đăng xuất khỏi hệ thống
- Giao diện hệ thống quay về màn hình đăng nhập, sẵn sàng cho phiên làm việc mới

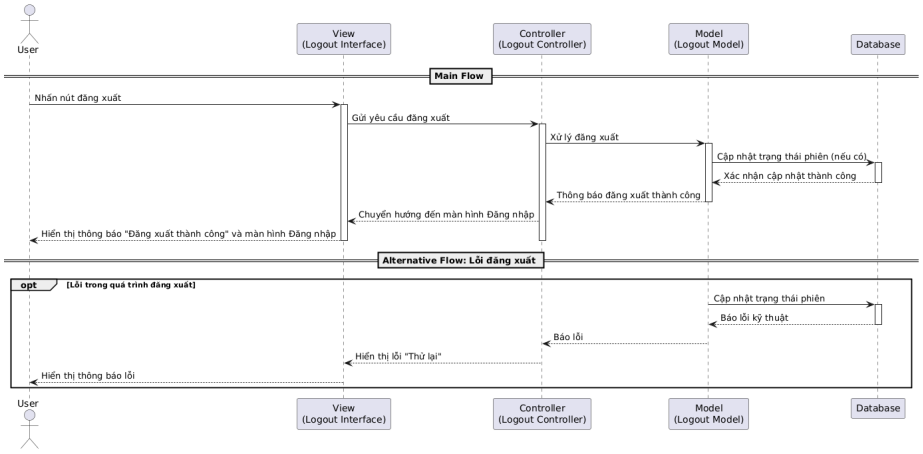

Figure 11: Biểu đồ tuần tự cho ca sử dụng đăng xuất tài khoản người dùng

6.2.3 Xem thông tin cá nhân của User

Name: Xem thông tin cá nhân mình
Actor: User
Pre-condition:

- User đã đăng nhập vào hệ thống
- Hệ thống hiển thị giao diện chính dành cho User

Main Flow:

1. User nhấn vào nút "Xem thông tin cá nhân"
2. Hệ thống truy xuất và hiển thị toàn bộ thông tin cá nhân của User, bao gồm:
    - Mã User
    - Họ tên đầy đủ
    - Email
    - Phòng ban
    - Đường dẫn hoặc hình ảnh khuôn mặt

Alternative Flow:

- **Bước 2:** Nếu có lỗi khi truy xuất thông tin cá nhân (ví dụ: lỗi hệ thống, lỗi kết nối database), hệ thống hiển thị thông báo lỗi và yêu cầu User thử lại sau

Post-condition:

- Hệ thống hiển thị thành công toàn bộ thông tin cá nhân của User đang đăng nhập

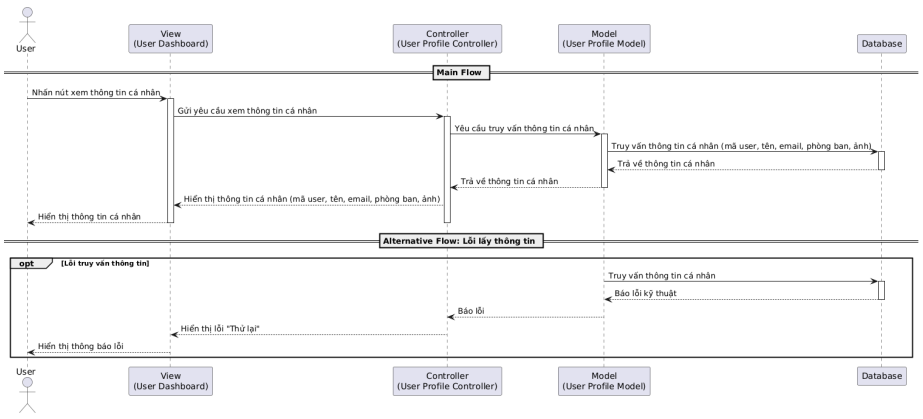

Figure 12: Biểu đồ tuần tự cho ca sử dụng người dùng xem thông tin của mình

6.2.4 Điểm danh bằng khuôn mặt

Name: Điểm danh bằng khuôn mặt
Actor: User
Pre-condition:

- User đã đăng nhập vào hệ thống
- Hệ thống hiển thị giao diện camera để thực hiện điểm danh bằng khuôn mặt

Main Flow:

1. User chụp ảnh khuôn mặt của mình thông qua camera trên giao diện điểm danh
2. Hệ thống xử lý ảnh và so sánh khuôn mặt được chụp với khuôn mặt đã lưu trong *database*
3. Nếu khuôn mặt trùng khớp, hệ thống ghi nhận điểm danh cho User vào ngày hôm đó
4. Hệ thống hiển thị thông báo điểm danh thành công và trả về thông tin chi tiết: mã điểm danh, ngày giờ, trạng thái

Alternative Flow:

- **Bước 1a:** Nếu User thoát ra hoặc chuyển hướng trang trong khi hệ thống đang xử lý ảnh, hệ thống sẽ huỷ thao tác và không ghi nhận điểm danh
- **Bước 1b:** Nếu ảnh chụp chứa nhiều hơn một khuôn mặt, hệ thống hiển thị cảnh báo và yêu cầu User chụp lại ảnh mới chỉ chứa một khuôn mặt
- **Bước 1c:** Nếu ảnh không rõ hoặc hệ thống không nhận dạng được khuôn mặt, hệ thống hiển thị lỗi và yêu cầu chụp lại ảnh rõ nét hơn
- **Bước 2a:** Nếu hệ thống không truy cập được đường dẫn ảnh, báo lỗi kỹ thuật và yêu cầu chụp lại
- **Bước 2b:** Nếu ảnh gốc trong *database* bị lỗi hoặc thiếu, hệ thống báo lỗi và yêu cầu liên hệ quản trị viên
- **Bước 3a:** Nếu User đã điểm danh trước đó trong cùng ngày, hệ thống thông báo "Đã điểm danh hôm nay" và không ghi nhận lại
- **Bước 3b:** Nếu có lỗi trong quá trình xác nhận điểm danh (ví dụ: lỗi ghi dữ liệu), hệ thống hiển thị lỗi và yêu cầu thử lại sau

Post-condition:

- User đã được điểm danh thành công bằng khuôn mặt cho ngày hiện tại
- Thông tin điểm danh được lưu vào *database* và có thể truy xuất khi cần

Figure 13: Biểu đồ tuần tự cho ca sử dụng người dùng điểm danh bằng khuôn mặt

6.2.5 Xem lịch sử điểm danh

Name: Xem lịch sử điểm danh
Actor: User
Pre-condition:

- User đã đăng nhập vào hệ thống
- Hệ thống hiển thị giao diện chính dành cho User

Main Flow:

1. User nhấn vào nút "Xem lịch sử điểm danh"
2. Hệ thống truy xuất và hiển thị danh sách lịch sử điểm danh của User, bao gồm các thông tin:
    - Mã điểm danh
    - Ngày, tháng, năm
    - Thời gian điểm danh
    - Trạng thái (đã điểm danh, chưa điểm danh, muộn, vắng,...)

Alternative Flow:

- **Bước 2:** Nếu có lỗi xảy ra trong quá trình truy xuất dữ liệu lịch sử điểm danh (ví dụ: lỗi kết nối cơ sở dữ liệu, lỗi hệ thống), hệ thống hiển thị thông báo lỗi và yêu cầu thử lại sau

Post-condition:

- User xem được đầy đủ thông tin lịch sử điểm danh của bản thân
- Giao diện cho phép User theo dõi, kiểm tra và tra cứu dữ liệu điểm danh của mình một cách thuận tiện

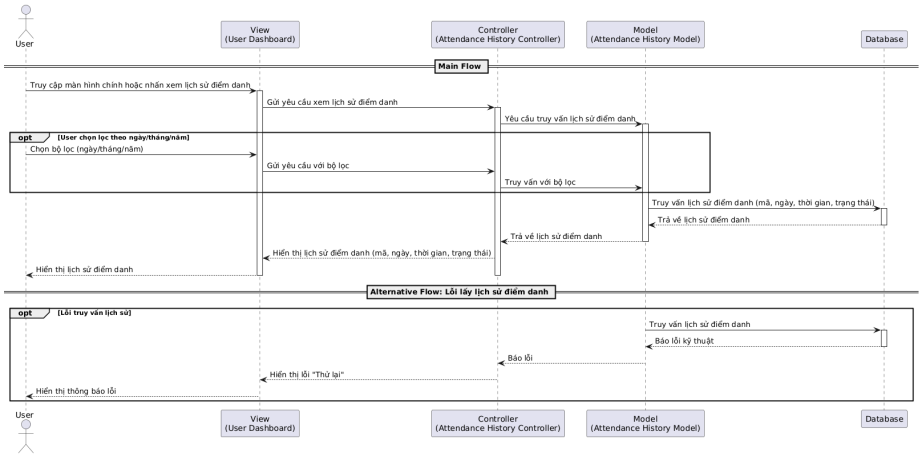

Figure 14: Biểu đồ tuần tự cho ca sử dụng người dùng xem lịch sử điểm danh

7 Thiết kế API

Để đảm bảo khả năng truy xuất dữ liệu nhanh chóng, thuận tiện và phục vụ đa dạng các nhu cầu từ phía client, hệ thống được xây dựng dựa trên kiến trúc **RESTful API**, sử dụng **FastAPI** – một framework hiện đại, hiệu suất cao của Python.
FastAPI được lựa chọn nhờ các ưu điểm nổi bật:

- Cấu trúc đơn giản, cú pháp rõ ràng, dễ tiếp cận và phát triển.
- Hiệu năng cao, hỗ trợ xử lý bất đồng bộ (`async/await`), giúp hệ thống hoạt động mượt mà ngay cả khi có nhiều yêu cầu đồng thời.
- Tương thích tốt với Tortoise ORM, giúp việc truy vấn đến cơ sở dữ liệu PostgreSQL trở nên thuận tiện và rõ ràng.
- Tích hợp sẵn Swagger UI tại địa chỉ `http://127.0.0.1:8000/docs`, hỗ trợ:
  - Kiểm thử trực tiếp các endpoint.
  - Theo dõi luồng dữ liệu dễ dàng.
  - Hỗ trợ hiệu quả trong quá trình debug và tài liệu hóa hệ thống.

Các nhóm API chính trong hệ thống điểm danh bao gồm:

1. **Nhóm xác thực (Authentication)**
   - `POST /auth/login` – Đăng nhập và trả về JWT token.
   - `POST /auth/logout` – Đăng xuất khỏi hệ thống.
2. **Nhóm người dùng (User)**
   - `GET /users/` – Lấy thông tin cá nhân người dùng hiện tại.
   - `POST /users/attendance` – Gửi ảnh khuôn mặt để thực hiện điểm danh.
   - `GET /users/attendance` – Xem lịch sử điểm danh của bản thân.
3. **Nhóm quản trị (Admin)**
   - `GET /admins/` – Xem danh sách tất cả người dùng.
   - `POST /admins/` – Tạo mới một tài khoản người dùng.
   - `PUT /admin/{user_id}` – Cập nhật thông tin người dùng theo ID.
   - `DELETE /admin/{user_id}` – Xóa một người dùng theo ID.
   - `GET /admins/attendance` – Xem tổng quan lịch sử điểm danh theo ngày.
   - `GET /admin/attendance/{user_id}` – Xem lịch sử điểm danh chi tiết của một người dùng cụ thể.

Cấu trúc API được tổ chức rõ ràng theo vai trò và chức năng, giúp việc mở rộng và bảo trì hệ thống trở nên dễ dàng hơn trong tương lai.

8 Triển khai và vận hành

8.1 Môi trường triển khai

Môi trường phát triển (Development):

- Sử dụng máy cá nhân hoặc server cục bộ.
- Chạy server backend FastAPI bằng `Uvicorn`.
- Frontend React chạy qua `npm dev server`.
- Sử dụng dịch vụ **Neon** miễn phí.

Môi trường kiểm thử (Staging/Test):

- Cài đặt trên máy chủ riêng.
- Kiểm thử tích hợp giữa frontend, backend và cơ sở dữ liệu.
- Dữ liệu kiểm thử mô phỏng hoạt động thực tế, nhưng không dùng dữ liệu thật.

Môi trường sản xuất (Production):

- Hệ thống được triển khai trên cloud hoặc server nội bộ.
- Backend được chạy bằng `Uvicorn/Gunicorn` kết hợp với **reverse proxy (Nginx)**.
- Frontend build và deploy tĩnh trên các dịch vụ như **Vercel**, **Netlify** hoặc host nội bộ.
- Cơ sở dữ liệu **PostgreSQL** sử dụng dịch vụ **Neon** hoặc host riêng có bảo mật cao.

8.2 Công cụ và dịch vụ đi kèm

- **Backend:** FastAPI (Python)
- **Frontend:** React
- **Cơ sở dữ liệu:** Neon (PostgreSQL)
- **ORM:** Tortoise ORM – quản lý model và truy vấn
- **Xác thực:** JWT – bảo mật API và phân quyền người dùng
- **Nhận diện khuôn mặt:** DeepFace

8.3 Chiến lược backup, giám sát và bảo trì

Backup dữ liệu:

- Tự động sao lưu cơ sở dữ liệu định kỳ (hằng ngày hoặc theo tuần).
- Lưu trữ backup tại các dịch vụ cloud an toàn.

Giám sát hệ thống:

- Theo dõi log của server backend.
- Giám sát thời gian phản hồi của API.
- Theo dõi hiệu suất nhận diện khuôn mặt nếu có nhiều ảnh cùng lúc.

Bảo trì hệ thống:

- Cập nhật định kỳ các dependency (`FastAPI`, `React`, `DeepFace`,...).
- Kiểm tra các bản vá bảo mật cho framework và thư viện.
- Làm sạch dữ liệu (ảnh cũ, người dùng không còn hoạt động) để tối ưu hệ thống.

9 Hiệu suất và khả năng mở rộng hệ thống

9.1 Hiệu suất hệ thống

Hệ thống được thiết kế nhằm đảm bảo hiệu suất tốt trong môi trường doanh nghiệp vừa và nhỏ, với các đặc điểm sau:

- **Xử lý điểm danh nhanh:** Nhận diện khuôn mặt trung bình <3 giây nhờ sử dụng mô hình AI **DeepFace**.
- **Tối ưu truy vấn dữ liệu:**
  - Sử dụng chỉ mục (`index`) trên các cột `user_id`, `checkin_time` để truy vấn lịch sử nhanh chóng.
- **Bảo mật hiệu suất cao:**
  - Sử dụng mã hóa mật khẩu (`bcrypt/scrypt`).
  - API được xác thực bằng **JWT token** (hoặc session/token-based auth).
- **Tải nhẹ trên frontend:**
  - Giao diện đơn giản, tải nhanh, phù hợp sử dụng nội bộ công ty.

9.2 Khả năng mở rộng

Hệ thống có khả năng mở rộng cả về người dùng, số lượng điểm danh và tính năng, với thiết kế linh hoạt:

1. Mở rộng về người dùng và dữ liệu
    - Backend tách rời frontend → có thể thay thế hoặc nâng cấp từng thành phần mà không ảnh hưởng toàn hệ thống.
    - Lưu trữ ảnh/vectơ đặc trưng riêng biệt → dễ tích hợp hệ thống lưu trữ phân tán (như `AWS S3`).
    - Cơ sở dữ liệu quan hệ có thể mở rộng theo chiều dọc hoặc phân mảnh nếu số lượng bản ghi lớn.
2. Mở rộng tính năng
    - Điểm danh qua **QR code** hoặc mã **OTP**.
    - Phân tích thống kê chuyên sâu (tổng hợp đi muộn, báo cáo theo tháng,...).
    - Gửi thông báo tự động (*email*/*slack*) nếu nhân viên chưa điểm danh.
3. Mở rộng theo kiến trúc
   Có thể triển khai theo mô hình **Microservices** nếu hệ thống phát triển lớn hơn, ví dụ:
    - Service nhận diện khuôn mặt riêng.
    - Service báo cáo & thống kê riêng.