

# Sequence Diagram

🕒 Created	@November 12, 2025 10:03 AM
🏷️ Tags	

## Usecase 1:

@startuml

title Sequence Diagram: UC-01 - Log in to the System (with Roles)

actor User as Actor

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Actor → FE: submitLoginForm(email, password)

activate FE

FE → BE: POST /api/token \n (body: {username: email, password: password})

activate BE

BE → DB: SELECT id, hashed\_password, role FROM users WHERE email = :email

activate DB

DB → BE: User record (including role)

deactivate DB

alt Login Successful

BE → BE: verifyPassword(password, user.hashed\_password)

' Giả sử verify thành công

BE → BE: createAccessToken(user.id, user.role)

note right: Payload của token giờ sẽ chứa cả vai trò (role)

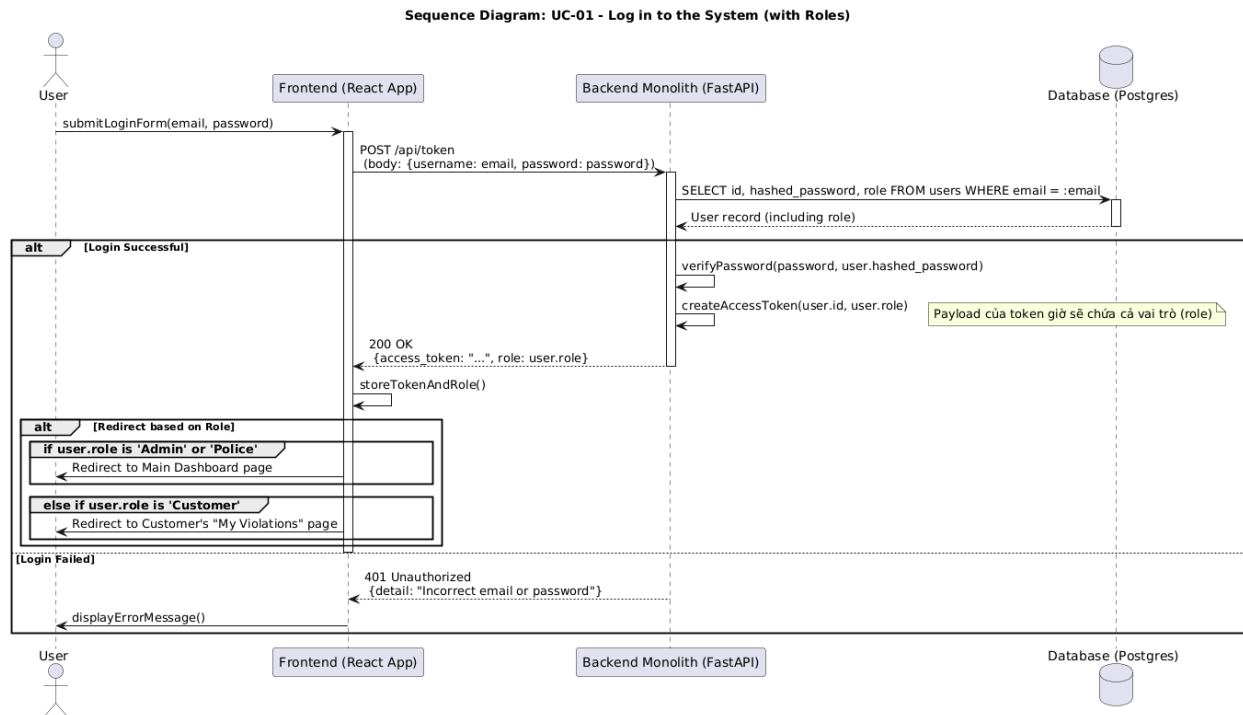
```
BE → FE: 200 OK \n {access_token: "...", role: user.role}
deactivate BE

FE → FE: storeTokenAndRole()

alt Redirect based on Role
  group if user.role is 'Admin' or 'Police'
    FE → Actor: Redirect to Main Dashboard page
  end
  group else if user.role is 'Customer'
    FE → Actor: Redirect to Customer's "My Violations" page
  end
end
deactivate FE
else Login Failed
  BE → FE: 401 Unauthorized \n {detail: "Incorrect email or password"}
  deactivate BE

  FE → Actor: displayErrorMessage()
  deactivate FE
end

@enduml
```



1. **User** (có thể là Admin, Police, hoặc Customer) nhập thông tin và nhấn nút Đăng nhập.
2. **Frontend** gửi request POST đến endpoint `/api/token` của **Backend**.
3. **Backend** nhận request, truy vấn **Database** để tìm người dùng có email tương ứng. **(Điểm mới)** Câu lệnh SELECT bây giờ sẽ lấy thêm cả cột role.
4. **Database** trả về bản ghi người dùng, bao gồm cả vai trò của họ.
5. **Backend** so sánh mật khẩu.
6. **(Luồng thành công)** Nếu mật khẩu khớp:
  - **(Điểm mới)** Backend tạo một JWT access token, và **nhúng thông tin role vào trong payload của token**.
  - **(Điểm mới)** Backend trả về cho Frontend không chỉ access\_token mà còn cả role của người dùng.
7. **Frontend** nhận được token và vai trò, lưu cả hai thông tin này lại (vào Local Storage hoặc Context).
8. **Frontend** thực hiện chuyển hướng có điều kiện:

- Nếu role là 'Admin' hoặc 'Police', chuyển hướng đến Dashboard giám sát chính.
  - Nếu role là 'Customer', chuyển hướng đến trang xem vi phạm cá nhân của họ.
9. **(Luồng thất bại)** Nếu mật khẩu không khớp, luồng xử lý vẫn như cũ: trả về lỗi 401 Unauthorized.

## Usecase 2:

@startuml

title Sequence Diagram: UC-02 - View Real-time Monitoring Dashboard (with Data Authorization)

actor "User (Admin/Police)" as Actor

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

participant "AI Processing Task\n(Background Process)" as AITask

' --- Step 1: Initial Page Load and WebSocket Connection ---

Actor → FE: navigateToDashboard()

activate FE

FE → BE: Establish WebSocket Connection \n (ws://.../ws/monitoring?token=jwt\_token)

activate BE

BE → BE: validateTokenAndRole(token)

note right: Kiểm tra token hợp lệ và vai trò \nphải là 'Admin' hoặc 'Police'. \nNếu không sẽ từ chối kết nối.

alt User is 'Police'

BE → DB: SELECT camera\_id FROM police\_camera\_assignments WHERE user\_id = :user\_id\_from\_token

activate DB

DB → BE: List of assigned\_camera\_ids

deactivate DB

BE → BE: storeInWebSocketSession(assigned\_camera\_ids)

note right: Lưu danh sách camera được phép \nxem vào session của WebSocket này.

end

BE → FE: WebSocket Connection Established

deactivate BE

FE → Actor: Display Dashboard Layout

deactivate FE

' --- Step 2: Real-time Data Pushing (Continuous Loop) ---

' Giả sử AI task đang chạy song song và đẩy dữ liệu vào Queue nội bộ

' AI Task → Queue: put({camera\_id: 1, ...})

' AI Task → Queue: put({camera\_id: 4, ...})

loop For each connected WebSocket client

BE → BE: getEventFromInternalQueue()

' Giả sử nhận được 1 event từ camera\_id: 1

alt Client is 'Admin'

' Admin được xem tất cả, không cần lọc

BE → FE: push(event\_data) via WebSocket

else Client is 'Police'

BE → BE: checkEventVsSession(event.camera\_id, session.assigned\_camera\_ids)

alt Event is from an assigned camera

```

        ' event.camera_id (1) nằm trong danh sách được gán
        BE → FE: push(event_data) via WebSocket
    else Event is NOT from an assigned camera
        ' event.camera_id (ví dụ là 4) không nằm trong ds được gán
        ' ⇒ Bỏ qua, không gửi cho client này.
    end
end
end
end

@enduml

```

## Phần 1: Thiết lập Kết nối (Initial Connection)

1. **User** (Admin/Police) truy cập vào trang Dashboard.
2. **Frontend** khởi tạo một kết nối WebSocket đến **Backend**. Nó sẽ gửi JWT token theo (có thể qua query parameter hoặc một cơ chế khác) để Backend có thể xác thực.
3. **Backend** nhận yêu cầu kết nối, giải mã JWT token để lấy user\_id và role. Nó kiểm tra xem role có phải là Admin hoặc Police không. Nếu là Customer, nó sẽ từ chối kết nối ngay lập tức.
4. **(Luồng dành cho Police)** Nếu role là Police, **Backend** sẽ thực hiện một truy vấn đến **Database** (bảng police\_camera\_assignments) để lấy danh sách các camera\_id mà người cảnh sát này được phép xem.
5. **Backend** sẽ lưu danh sách assigned\_camera\_ids này vào trong "phiên" (session) của chính kết nối WebSocket đó. Mỗi client kết nối sẽ có một session riêng.
6. Sau khi hoàn tất, **Backend** xác nhận kết nối thành công, và **Frontend** hiển thị giao diện Dashboard.

## Phần 2: Luồng Đẩy Dữ liệu Thời gian thực (Continuous Pushing)

Phần này diễn ra liên tục sau khi kết nối đã được thiết lập.

1. Bên trong **Backend**, một luồng nền liên tục lấy các sự kiện (frame ảnh hoặc vi phạm) từ Queue nội bộ (do AI Task đẩy vào). Mỗi sự kiện này đều chứa thông tin camera\_id nguồn.

2. **Backend** duyệt qua tất cả các client đang kết nối qua WebSocket. Với mỗi client, nó sẽ kiểm tra:

- **Nếu client là Admin:** Không cần kiểm tra gì thêm, **Backend** ngay lập tức đẩy (push) sự kiện đó qua WebSocket cho Admin.
- **Nếu client là Police:**
  - **(Điểm mới)** **Backend** sẽ lấy danh sách `assigned_camera_ids` đã lưu trong session của client đó (ở Phần 1).
  - Nó so sánh `camera_id` của sự kiện với danh sách này.
  - Nếu `camera_id` của sự kiện **có** trong danh sách được phép, nó sẽ đẩy sự kiện đó cho client Police.
  - Nếu `camera_id` của sự kiện **không có** trong danh sách, nó sẽ **bỏ qua** và không làm gì cả đối với client này.

**Kết quả:** Một Admin sẽ thấy dữ liệu từ tất cả các camera, trong khi một Police chỉ thấy dữ liệu từ những camera đã được gán cho anh ta, đảm bảo an toàn và phân quyền dữ liệu ngay từ tầng giao tiếp thời gian thực.

---

## Usecase 3:

@startuml

title Sequence Diagram: UC-03 - Admin Manages System Configuration (Creates a Rule)

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: submitNewRuleForm(name, dsl\_content)

activate FE

FE → BE: POST /api/rules \n (Header: Authorization: Bearer jwt\_token) \n (Body: {name: "...", content: "..."})

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Middleware kiểm tra token và xác thực \nvai trò người dùng phải là 'Admin'.

alt User is Admin AND DSL Syntax is Valid

BE → BE: DslParser.parse(dsl\_content)

' Giả sử parse thành công

BE → DB: INSERT INTO rules (name, dsl\_content, created\_by\_id) VALUES (...)

activate DB

DB → BE: New rule\_id

deactivate DB

BE → FE: 201 Created \n {id: rule\_id, ...}

deactivate BE

FE → Admin: displaySuccessMessage("Rule created successfully!")

deactivate FE

else User is NOT Admin OR DSL Syntax is Invalid

alt User is NOT Admin

BE → FE: 403 Forbidden \n {detail: "Admin privileges required"}

else DSL Syntax is Invalid

BE → FE: 400 Bad Request \n {detail: "Syntax error on line X..."}

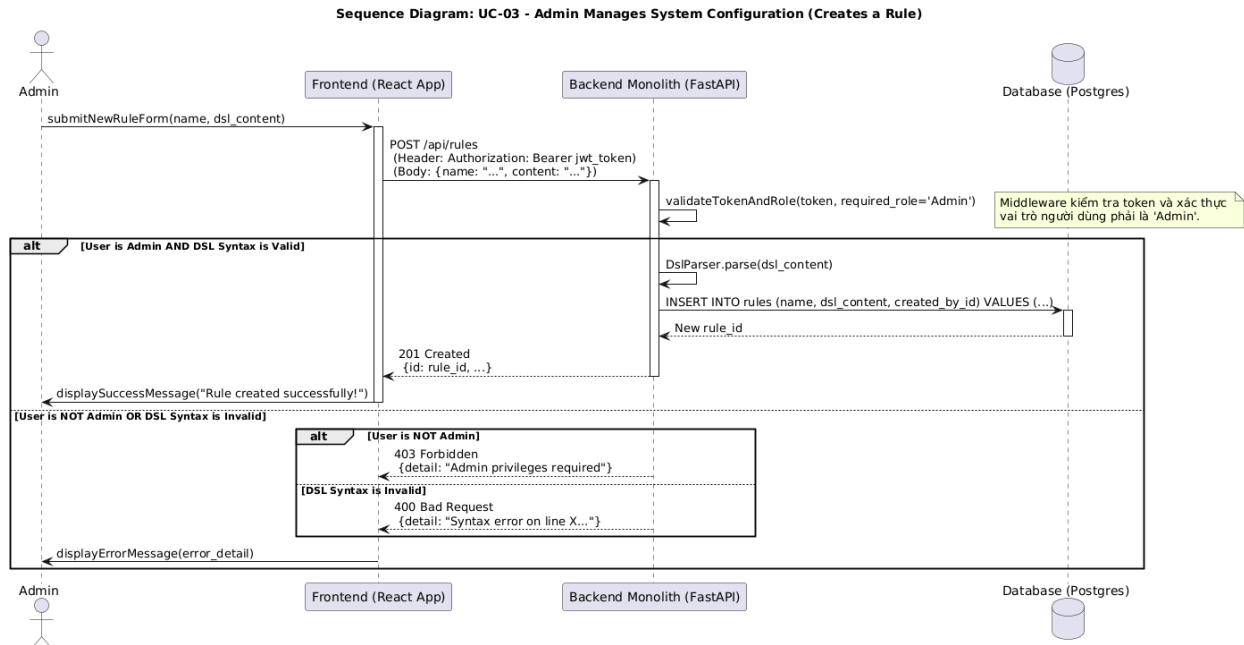
end

deactivate BE

FE → Admin: displayErrorMessage(error\_detail)

deactivate FE

end



1. **Admin** vào trang quản lý, điền tên và nội dung DSL cho quy tắc mới, sau đó nhấn Lưu.
2. **Frontend** gửi request POST /api/rules đến **Backend**, đính kèm JWT token.
3. **Backend** nhận request và thực hiện **kiểm tra kép**:
  - a. Token có hợp lệ không?
  - b. Vai trò (role) trong token có phải là Admin không?
4. **(Luồng thành công)** Nếu đúng là Admin, **Backend** tiếp tục gọi module **DslParser** để kiểm tra cú pháp của dsl\_content.
  - Nếu cú pháp hợp lệ, **Backend** lưu quy tắc mới vào **Database** (bảng rules) và trả về 201 Created.
  - **Frontend** nhận tín hiệu thành công và cập nhật giao diện.
5. **(Luồng thất bại)**
  - Nếu người dùng không phải Admin, **Backend** trả về lỗi 403 Forbidden.

- Nếu người dùng là Admin nhưng cú pháp DSL sai, **Backend** trả về lỗi 400 Bad Request kèm thông tin chi tiết.
- **Frontend** hiển thị thông báo lỗi tương ứng.

## Usecase 4:

@startuml

title Sequence Diagram: UC-04 - Assign Camera to Police

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Navigate to "Camera Assignment" page

activate FE

' --- Step 1: Frontend fetches necessary data for the UI ---

FE → BE: GET /api/users?role=police \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → FE: 200 OK \n {users: [police\_user\_1, ...]}

deactivate BE

FE → BE: GET /api/cameras \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → FE: 200 OK \n {cameras: [camera\_1, ...]}

deactivate BE

FE → Admin: Display dropdowns/lists of Police and Cameras

deactivate FE

' --- Step 2: Admin performs the assignment ---

Admin → FE: Selects a Police user and multiple Cameras

Admin → FE: Clicks "Save Assignment" button  
activate FE

FE → BE: POST /api/assignments \n (Header: Authorization: Bearer jwt\_token)  
\n (Body: {user\_id: ..., camera\_ids: [...]}))  
activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')  
note right: Xác thực quyền Admin

alt User is Admin AND Input is Valid

' Backend có thể cần xóa các assignment cũ của user này trước khi tạo mới

BE → DB: DELETE FROM police\_camera\_assignments WHERE user\_id = :user\_id  
er\_id

activate DB

DB → BE: (Deletion complete)

deactivate DB

' Lặp qua danh sách camera\_ids để tạo các bản ghi mới

loop for each camera\_id in camera\_ids

BE → DB: INSERT INTO police\_camera\_assignments (user\_id, camera\_id)  
VALUES (...)

activate DB

DB → BE: (Insertion complete)

deactivate DB

end

BE → FE: 200 OK \n {message: "Assignment updated successfully"}  
deactivate BE

FE → Admin: displaySuccessMessage("Assignment saved!")

deactivate FE

else User is NOT Admin OR Input is Invalid

alt User is NOT Admin

BE → FE: 403 Forbidden

else Input is Invalid (e.g., user\_id or camera\_id does not exist)

BE → FE: 404 Not Found

end

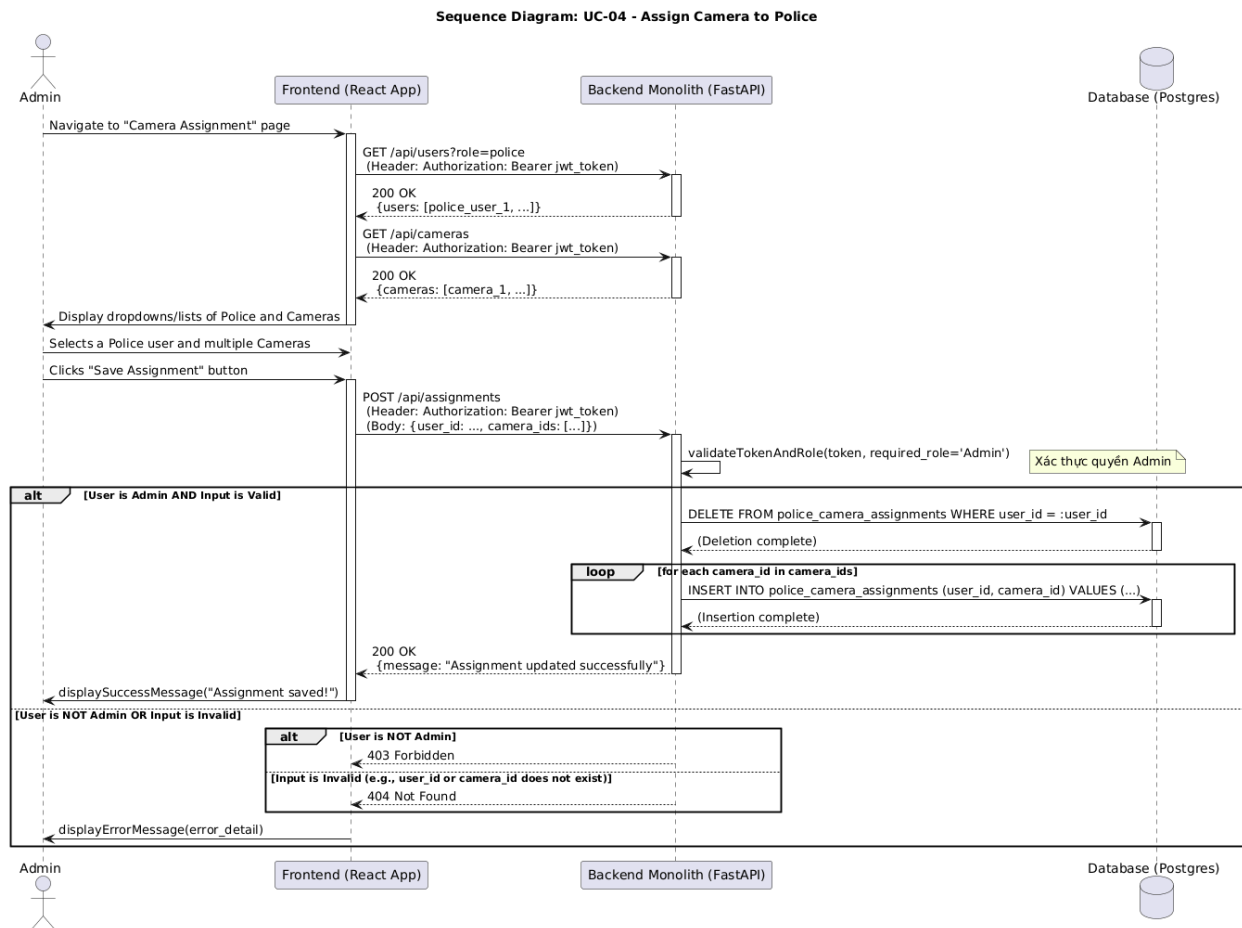
deactivate BE

FE → Admin: displayErrorMessage(error\_detail)

deactivate FE

end

@enduml



Sơ đồ này được chia làm hai phần chính:

## Phần 1: Tài dữ liệu cho Giao diện

### 1. Admin truy cập trang "Gán Camera".

2. Để xây dựng giao diện, **Frontend** cần dữ liệu. Nó sẽ thực hiện hai lời gọi API song song đến **Backend**:
  - GET /api/users?role=police để lấy danh sách tất cả người dùng có vai trò là Police.
  - GET /api/cameras để lấy danh sách tất cả các camera hiện có.
3. **Backend** trả về dữ liệu và **Frontend** hiển thị các danh sách này (ví dụ: dưới dạng dropdown hoặc multi-select list) cho **Admin**.

## Phần 2: Thực hiện Gán quyền

1. **Admin** chọn một Police và một (hoặc nhiều) Camera, sau đó nhấn nút Lưu.
2. **Frontend** đóng gói các ID đã chọn (user\_id và một mảng camera\_ids) và gửi một request POST đến một endpoint mới, ví dụ /api/assignments.
3. **Backend** nhận request và đầu tiên, **kiểm tra quyền Admin**.
4. **(Luồng thành công)** Nếu là Admin và dữ liệu hợp lệ:
  - **Backend** thực hiện logic cập nhật trong **Database**. Một cách tiếp cận phổ biến và an toàn là:
    - a. Xóa tất cả các bản ghi gán quyền cũ của user\_id đó trong bảng police\_camera\_assignments.
    - b. Lặp qua danh sách camera\_ids mới và INSERT từng bản ghi gán quyền mới vào.

*Điều này đảm bảo rằng danh sách gán quyền luôn là danh sách mới nhất mà Admin đã chọn.*
  - Sau khi cập nhật DB thành công, **Backend** trả về 200 OK.
  - **Frontend** hiển thị thông báo thành công.
5. **(Luồng thất bại)** Nếu người dùng không phải Admin, hoặc user\_id/camera\_id không tồn tại, **Backend** sẽ trả về lỗi 403 (Forbidden) hoặc 404 (Not Found), và **Frontend** sẽ thông báo lỗi cho Admin.

---

## Usecase 5:

@startuml

title Sequence Diagram: UC-05 - View and Filter Violation Reports for Police

actor Police

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Police → FE: navigateToReportsPage()

activate FE

' --- Initial Load ---

FE → BE: GET /api/violations \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndGetUserId(token)

note right: Giải mã token để lấy user\_id

' --- Data Authorization Step ---

BE → DB: SELECT camera\_id FROM police\_camera\_assignments WHERE user\_id = :user\_id

activate DB

DB → BE: List of assigned\_camera\_ids

deactivate DB

alt Police is assigned to some cameras

' --- Data Fetching Step ---

BE → DB: SELECT \* FROM violations \n WHERE camera\_id IN :assigned\_camera\_ids \n ORDER BY timestamp DESC LIMIT 20

activate DB

DB → BE: List of recent, authorized violations

deactivate DB

BE → FE: 200 OK \n {violations: [...]}

deactivate BE

```

    FE → Police: displayViolationsList(violations)
    deactivate FE
else Police has no assigned cameras
    BE → FE: 200 OK \n {violations: []}
    deactivate BE

    FE → Police: displayEmptyStateMessage("You have not been assigned to a
ny cameras.")
    deactivate FE
end

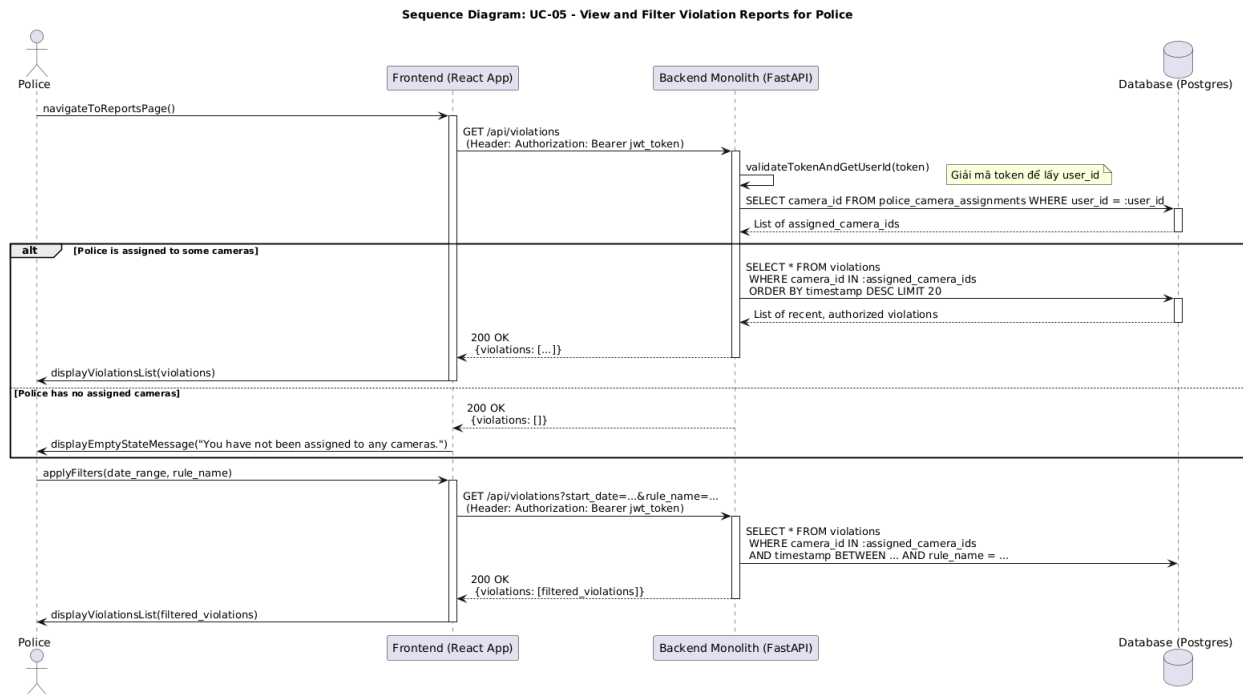
' --- Filtering (Tương tự, nhưng với điều kiện lọc thêm) ---
Police → FE: applyFilters(date_range, rule_name)
activate FE

FE → BE: GET /api/violations?start_date=...&rule_name=... \n (Header: Authori
zation: Bearer jwt_token)
activate BE
' Backend lặp lại các bước: validateToken → getAssignedCameras → queryWit
hFilters
BE → DB: SELECT * FROM violations \n WHERE camera_id IN :assigned_came
ra_ids \n AND timestamp BETWEEN ... AND rule_name = ...
' ... (phần còn lại tương tự)

BE → FE: 200 OK \n {violations: [filtered_violations]}
deactivate BE
FE → Police: displayViolationsList(filtered_violations)
deactivate FE

@enduml

```



1. **Police** sau khi đăng nhập, truy cập vào trang "Báo cáo".
2. **Frontend** gửi một request GET /api/violations đến **Backend**, đính kèm JWT token.
3. **Backend** nhận request và giải mã token để lấy user\_id.
4. **Bước Phân quyền Dữ liệu:**
  - **Backend** dùng user\_id này để truy vấn vào bảng trung gian police\_camera\_assignments.
  - **Database** trả về một danh sách các camera\_id mà người Police này được phép xem (ví dụ: [1, 2, 5]).
5. **Bước Lấy Dữ liệu:**
  - **(Luồng thành công)** Nếu danh sách assigned\_camera\_ids không rỗng, **Backend** tiếp tục thực hiện một truy vấn thứ hai vào bảng violations.
  - **(Điểm mới)** Câu lệnh SELECT này sẽ có một điều kiện WHERE cực kỳ quan trọng: camera\_id IN (1, 2, 5).
  - **Database** trả về danh sách các vi phạm chỉ từ những camera được phép.
  - **Backend** trả về danh sách này cho **Frontend**.

- **(Luồng không có quyền)** Nếu danh sách assigned\_camera\_ids rỗng, **Backend** sẽ trả về một mảng rỗng [] và **Frontend** sẽ hiển thị thông báo tương ứng.

6. **Frontend** hiển thị danh sách vi phạm đã được lọc sẵn cho **Police**.

## Usecase 6:

@startuml

title Sequence Diagram: UC-06 - View Violation Reports for Customer

actor Customer

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Customer → FE: navigateToMyViolationsPage()

activate FE

FE → BE: GET /api/me/violations \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndGetUserId(token, required\_role='Customer')

note right: Giải mã token, lấy user\_id và \nxác thực vai trò là 'Customer'.

' --- Data Authorization Step ---

BE → DB: SELECT license\_plate FROM users WHERE id = :user\_id

activate DB

DB → BE: customer\_license\_plate

deactivate DB

alt Customer has a registered license plate

' --- Data Fetching Step ---

BE → DB: SELECT \* FROM violations \n WHERE detected\_license\_plate = :customer\_license\_plate \n ORDER BY timestamp DESC

activate DB

DB → BE: List of personal violations

deactivate DB

BE → FE: 200 OK \n {violations: [...]}

deactivate BE

FE → Customer: displayPersonalViolations(violations)

deactivate FE

else Customer has NOT registered a license plate

BE → FE: 404 Not Found \n {detail: "License plate not found for this account"}

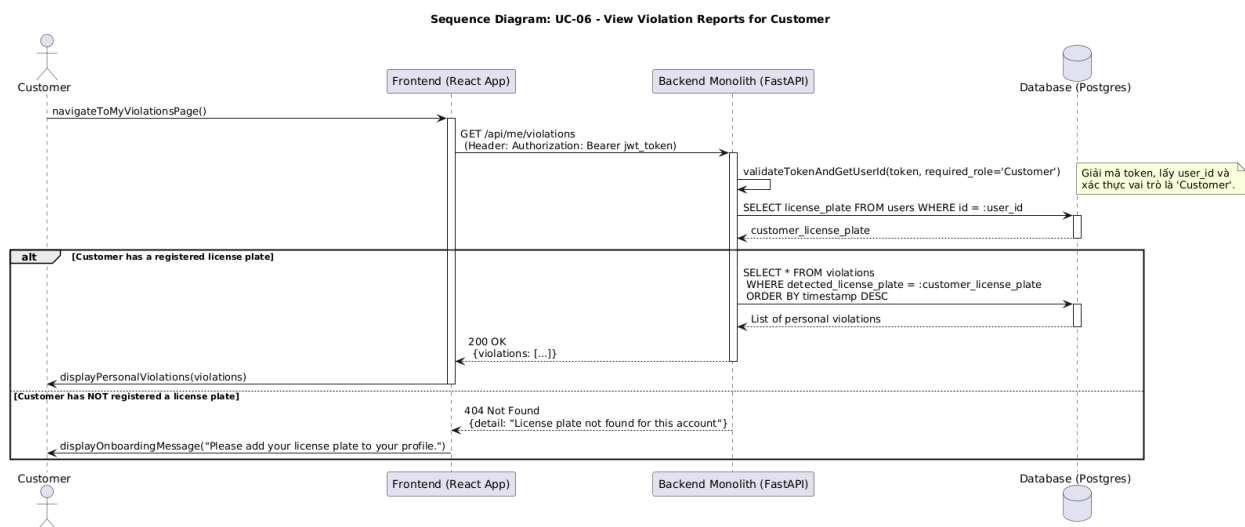
deactivate BE

FE → Customer: displayOnboardingMessage("Please add your license plate to your profile.")

deactivate FE

end

@enduml



1. **Customer** sau khi đăng nhập, truy cập trang "Vi phạm của tôi".
2. **Frontend** gửi request GET /api/me/violations đến **Backend**, đính kèm JWT token.
3. **Backend** giải mã token, lấy user\_id và xác nhận vai trò là Customer.
4. **Backend** dùng user\_id để truy vấn bảng users trong **Database** nhằm lấy ra license\_plate của Customer đó.
5. **(Luồng thành công)** Nếu license\_plate tồn tại:
  - **Backend** thực hiện truy vấn thứ hai vào bảng violations với điều kiện WHERE detected\_license\_plate = [biển số xe của customer].
  - **Database** trả về danh sách các vi phạm của chính khách hàng đó.
  - **Backend** gửi danh sách này cho **Frontend**.
6. **(Luồng thất bại)** Nếu tài khoản Customer chưa đăng ký biển số xe, **Backend** trả về lỗi 404 Not Found, và **Frontend** hiển thị thông báo hướng dẫn.
7. **Frontend** hiển thị kết quả cho **Customer**.

---

## Usecase 7:

---

### UseCase 7.1: Tạo một Tài khoản Người dùng mới (Create a New User Account)

@startuml

title Sequence Diagram: UC-07.1 - Admin Creates a New User Account (Police/Customer)

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Navigates to User Management and clicks "Create User"

Admin → FE: Fills and submits the new user form \n (email, password, role, license\_plate?)

activate FE

FE → BE: POST /api/users \n (Header: Authorization: Bearer jwt\_token) \n (Body: {email: ..., password: ..., role: ...})

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission

' --- Step 2: Validate the input data ---

BE → DB: SELECT id FROM users WHERE email = :email

activate DB

DB → BE: (Returns null if email is unique)

deactivate DB

alt Email is unique and data is valid

' --- Step 3: Create the user ---

BE → BE: hashPassword(password)

BE → DB: INSERT INTO users (email, hashed\_password, role, ...) VALUES (...)

activate DB

DB → BE: New user\_id

deactivate DB

BE → FE: 201 Created \n {id: user\_id, email: ..., role: ...}

deactivate BE

FE → Admin: displaySuccessMessage("User created successfully!")

deactivate FE

else Email already exists

BE → FE: 409 Conflict \n {detail: "Email already registered."}

deactivate BE

FE → Admin: displayErrorMessage("This email is already in use.")

deactivate FE

end

else Admin does NOT have permission

BE → FE: 403 Forbidden \n {detail: "Admin privileges required"}

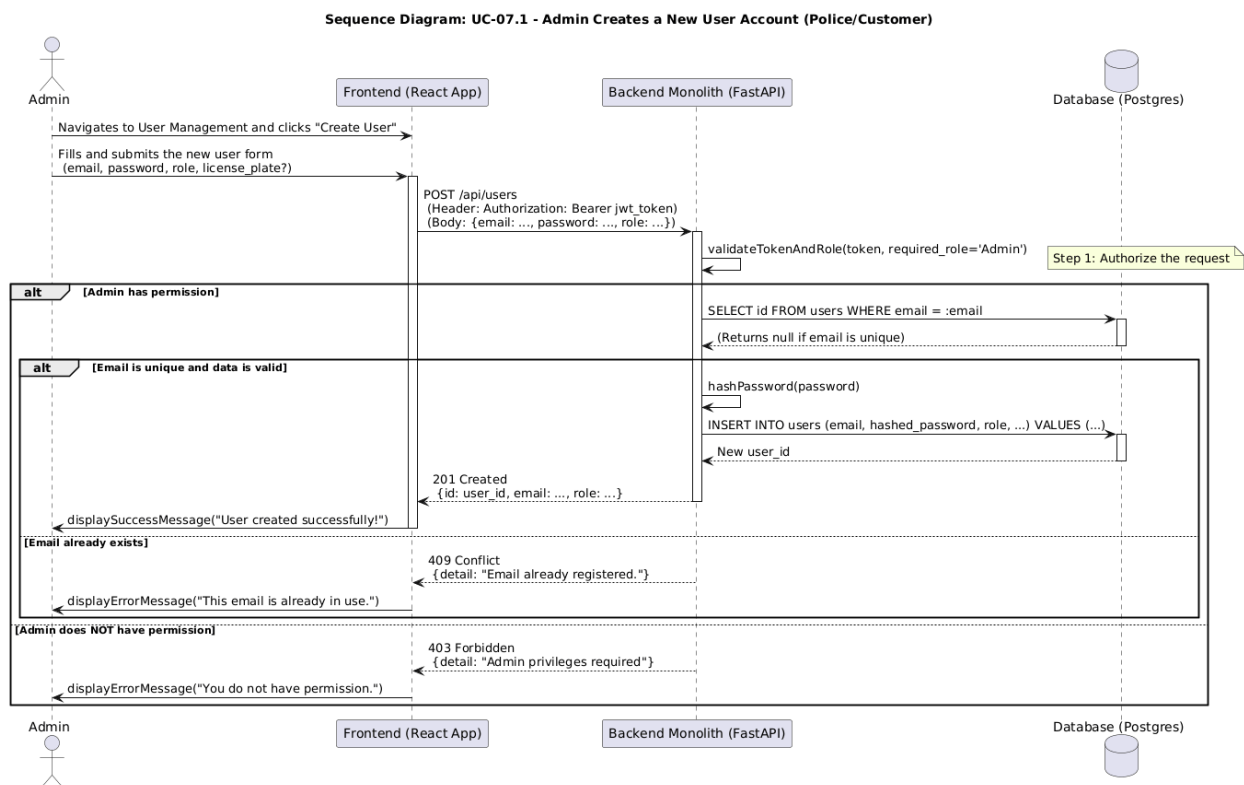
deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")

deactivate FE

end

@enduml



1. **Admin** vào trang quản lý người dùng, điền form tạo tài khoản mới (bao gồm cả việc chọn vai trò là Police hay Customer) và nhấn Lưu.
2. **Frontend** gửi request POST /api/users đến **Backend**, đính kèm JWT token.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** tiếp tục kiểm tra tính hợp lệ của dữ liệu, quan trọng nhất là email không được trùng.
5. Nếu dữ liệu hợp lệ, **Backend** băm mật khẩu và INSERT bản ghi mới vào bảng users trong **Database**.
6. **Backend** trả về 201 Created để báo hiệu tạo thành công.
7. **Frontend** hiển thị thông báo thành công cho **Admin**.

## Use Case 7.2: Xem Danh sách Người dùng (Read User Accounts)

@startuml

title Sequence Diagram: UC-07.2 - Admin Reads User Accounts

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Navigates to User Management page

activate FE

FE → BE: GET /api/users \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

```

alt Admin has permission
    ' --- Step 2: Fetch the data ---
    BE → DB: SELECT id, email, full_name, role, license_plate FROM users
    activate DB
    DB → BE: List of all users
    deactivate DB

    BE → FE: 200 OK \n {users: [...]}
    deactivate BE

    FE → Admin: displayUsersList(users)
    deactivate FE
else Admin does NOT have permission
    BE → FE: 403 Forbidden
    deactivate BE

    FE → Admin: displayErrorMessage("You do not have permission.")
    deactivate FE
end

' --- Optional: Filtering by role ---
Admin → FE: Clicks filter button (e.g., "Show only Police")
activate FE

FE → BE: GET /api/users?role=police \n (Header: Authorization: Bearer jwt_tok
en)
activate BE

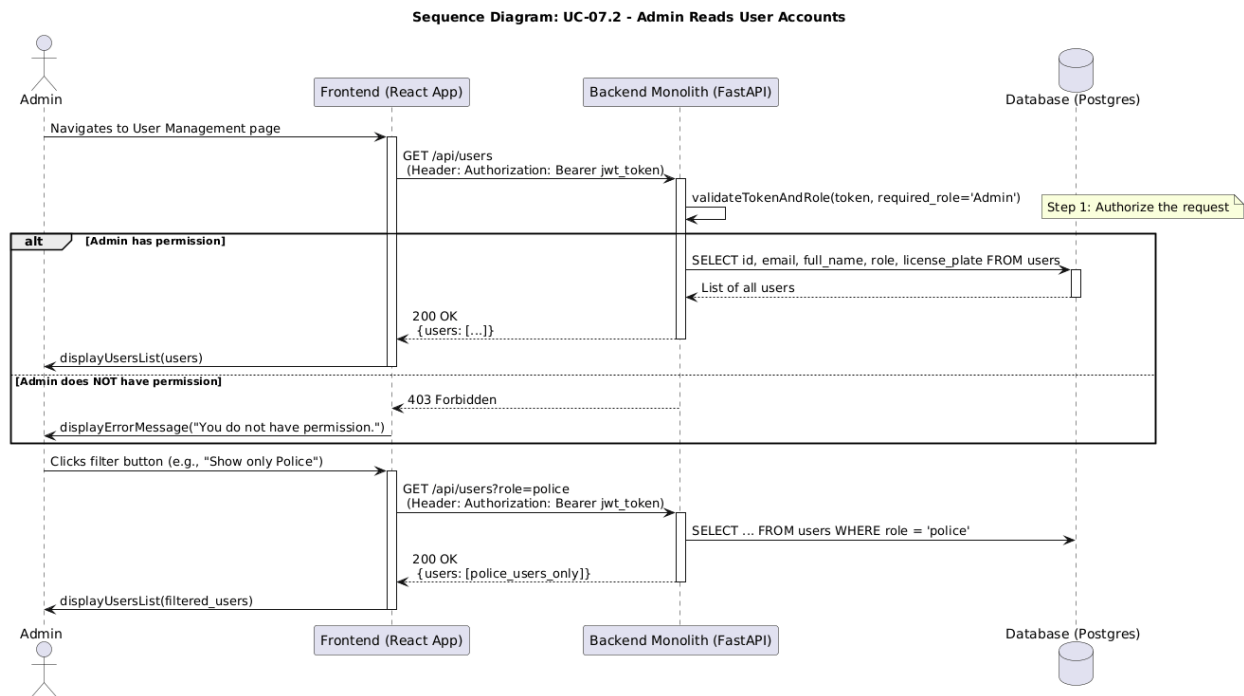
' Backend repeats the authorization and fetching process, but with a WHERE c
lause
BE → DB: SELECT ... FROM users WHERE role = 'police'
' ... (rest of the flow is similar) ...

BE → FE: 200 OK \n {users: [police_users_only]}
deactivate BE

```

FE → Admin: displayUsersList(filtered\_users)  
deactivate FE

@enduml



1. **Admin** truy cập vào trang "Quản lý Người dùng".
2. **Frontend** ngay lập tức gửi request GET /api/users đến **Backend**, đính kèm JWT token.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** truy vấn **Database** để lấy danh sách tất cả người dùng (hoặc một trang dữ liệu - pagination).
5. **Backend** trả về danh sách này cho **Frontend** với status 200 OK.
6. **Frontend** hiển thị danh sách người dùng trong một bảng.
7. **(Luồng lọc tùy chọn)** Khi Admin muốn lọc, ví dụ chỉ xem các Police:
  - **Frontend** gửi một request mới, ví dụ GET /api/users?role=police.

- **Backend** lặp lại quy trình nhưng thêm điều kiện WHERE role = 'police' vào câu truy vấn và trả về kết quả đã được lọc.

## Use Case 7.3: Cập nhật Thông tin Người dùng (Update a User Account)

@startuml

title Sequence Diagram: UC-07.3 - Admin Updates a User Account

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Clicks "Edit" on a user in the list

Admin → FE: Modifies user information in the form and submits

activate FE

FE → BE: PUT /api/users/{user\_id} \n (Header: Authorization: Bearer jwt\_token) \n (Body: {full\_name: ..., role: ..., ...})

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission

' --- Step 2: Validate the input data (if necessary) ---

' Ví dụ, nếu email được phép thay đổi, cần kiểm tra trùng lặp

' BE → DB: Check for email uniqueness if email is changed

alt Data is valid

' --- Step 3: Update the user ---

BE → DB: UPDATE users SET full\_name = ..., role = ... WHERE id = :user\_id

```

d
    activate DB
    DB → BE: (Update successful)
    deactivate DB

    BE → FE: 200 OK \n {updated_user_data}
    deactivate BE

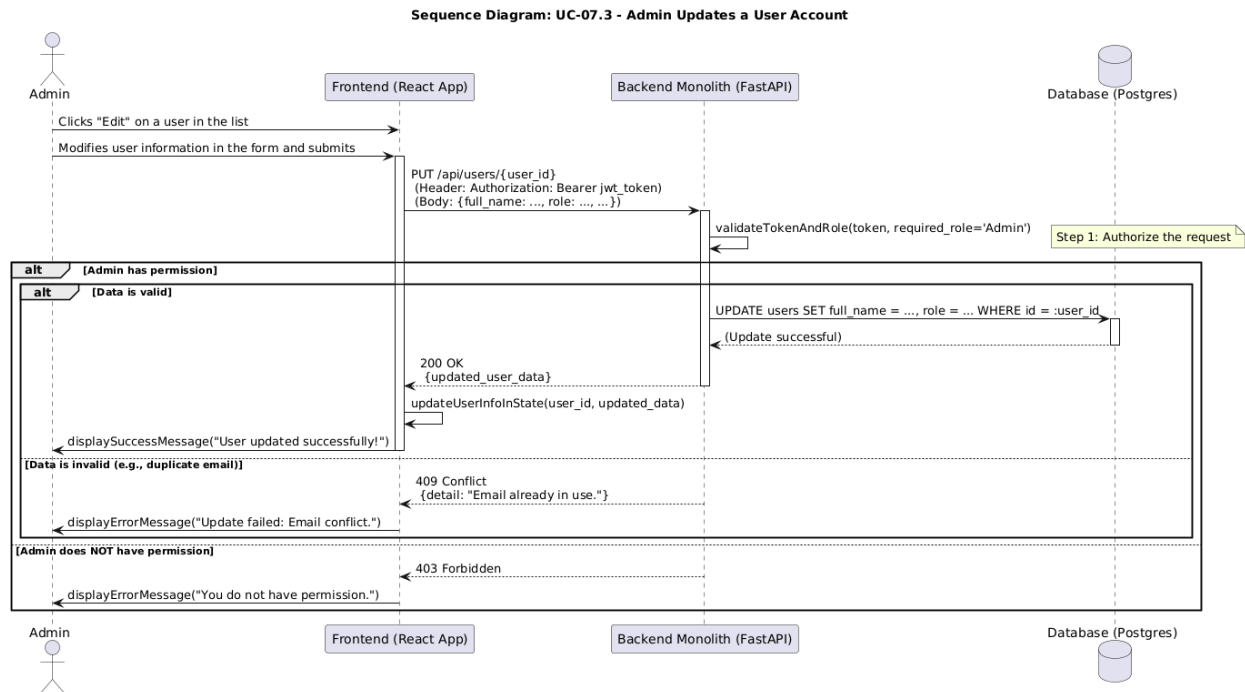
    FE → FE: updateUserInfoInState(user_id, updated_data)
    FE → Admin: displaySuccessMessage("User updated successfully!")
    deactivate FE
else Data is invalid (e.g., duplicate email)
    BE → FE: 409 Conflict \n {detail: "Email already in use."}
    deactivate BE

    FE → Admin: displayErrorMessage("Update failed: Email conflict.")
    deactivate FE
end
else Admin does NOT have permission
    BE → FE: 403 Forbidden
    deactivate BE

    FE → Admin: displayErrorMessage("You do not have permission.")
    deactivate FE
end

@enduml

```



1. **Admin** chọn một người dùng từ danh sách và nhấn "Sửa", sau đó thay đổi thông tin trong form và nhấn "Lưu thay đổi".
2. **Frontend** gửi request PUT (hoặc PATCH) đến endpoint `/api/users/{user_id}`, trong đó `{user_id}` là ID của người dùng cần sửa.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** tiếp tục xác thực dữ liệu mới (nếu cần).
5. Nếu dữ liệu hợp lệ, **Backend** thực hiện câu lệnh UPDATE trong **Database** để cập nhật bản ghi tương ứng.
6. **Backend** trả về 200 OK cùng với thông tin người dùng đã được cập nhật.
7. **Frontend** cập nhật lại giao diện và hiển thị thông báo thành công.

## Use Case 7.4: Xóa một Tài khoản Người dùng (Delete a User Account)

@startuml

title Sequence Diagram: UC-07.4 - Admin Deletes a User Account (Soft Delete)

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Clicks "Delete" on a user in the list

Admin → FE: Confirms the deletion in a popup dialog

activate FE

FE → BE: DELETE /api/users/{user\_id} \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission

' --- Step 2: Perform Soft Delete ---

' Thay vì DELETE, chúng ta sẽ UPDATE một cờ

BE → DB: UPDATE users SET is\_active = false WHERE id = :user\_id

activate DB

DB → BE: (Update successful)

deactivate DB

BE → FE: 204 No Content

note right: 204 là status code phù hợp cho hành động \nDELETE thành công mà không cần trả về body.

deactivate BE

FE → FE: removeUserFromListInState(user\_id)

FE → Admin: displaySuccessMessage("User deactivated successfully!")

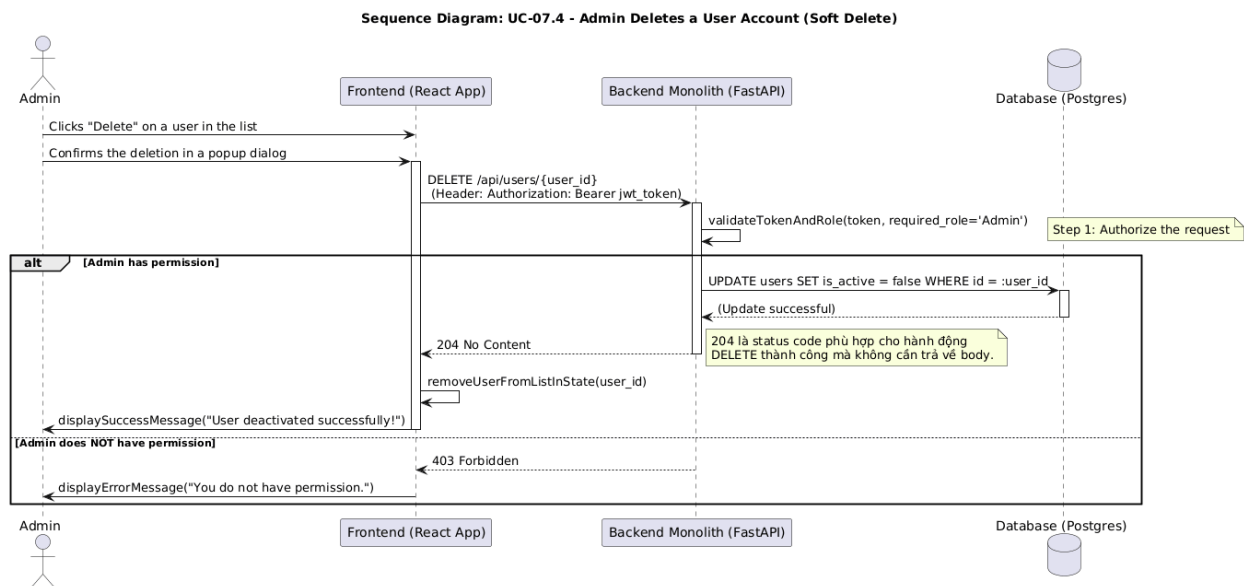
```

deactivate FE
else Admin does NOT have permission
  BE → FE: 403 Forbidden
  deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")
deactivate FE
end

@enduml

```



1. **Admin** nhấn nút "Xóa" bên cạnh một người dùng và xác nhận hành động.
2. **Frontend** gửi request DELETE đến endpoint `/api/users/{user_id}`.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** không thực hiện lệnh DELETE từ **Database**. Thay vào đó, nó thực hiện một lệnh UPDATE để cập nhật trạng thái của người dùng, ví dụ SET `is_active = false`.
5. **Backend** trả về status code 204 No Content, là một chuẩn RESTful để thông báo hành động xóa thành công.

6. **Frontend** nhận được phản hồi này, xóa người dùng đó khỏi danh sách trên giao diện và hiển thị thông báo.

## Use Case 7.5: Thêm một Camera mới (Create a New Camera)

@startuml

title Sequence Diagram: UC-07.5 - Admin Creates a New Camera

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Navigates to Camera Management page

Admin → FE: Fills and submits the new camera form (name, source\_url)

activate FE

FE → BE: POST /api/cameras \n (Header: Authorization: Bearer jwt\_token) \n (Body: {name: ..., source\_url: ...})

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission AND Input is Valid

' --- Step 2: Validate the input data ---

' Có thể kiểm tra xem source\_url có bị trùng không

BE → DB: SELECT id FROM cameras WHERE source\_url = :source\_url

activate DB

DB → BE: (Returns null if URL is unique)

deactivate DB

alt Source URL is unique

```

' --- Step 3: Create the camera record ---
BE → DB: INSERT INTO cameras (name, source_url, is_active) VALUES
(...)
activate DB
DB → BE: New camera_id
deactivate DB

' --- Step 4: Notify the AI processing module ---
BE → BE: notifyAIProcessingTaskOfChange()
note right: Gửi tín hiệu nội bộ để AI Task quét lại danh sách camera active.

BE → FE: 201 Created \n {id: camera_id, name: ..., ...}
deactivate BE

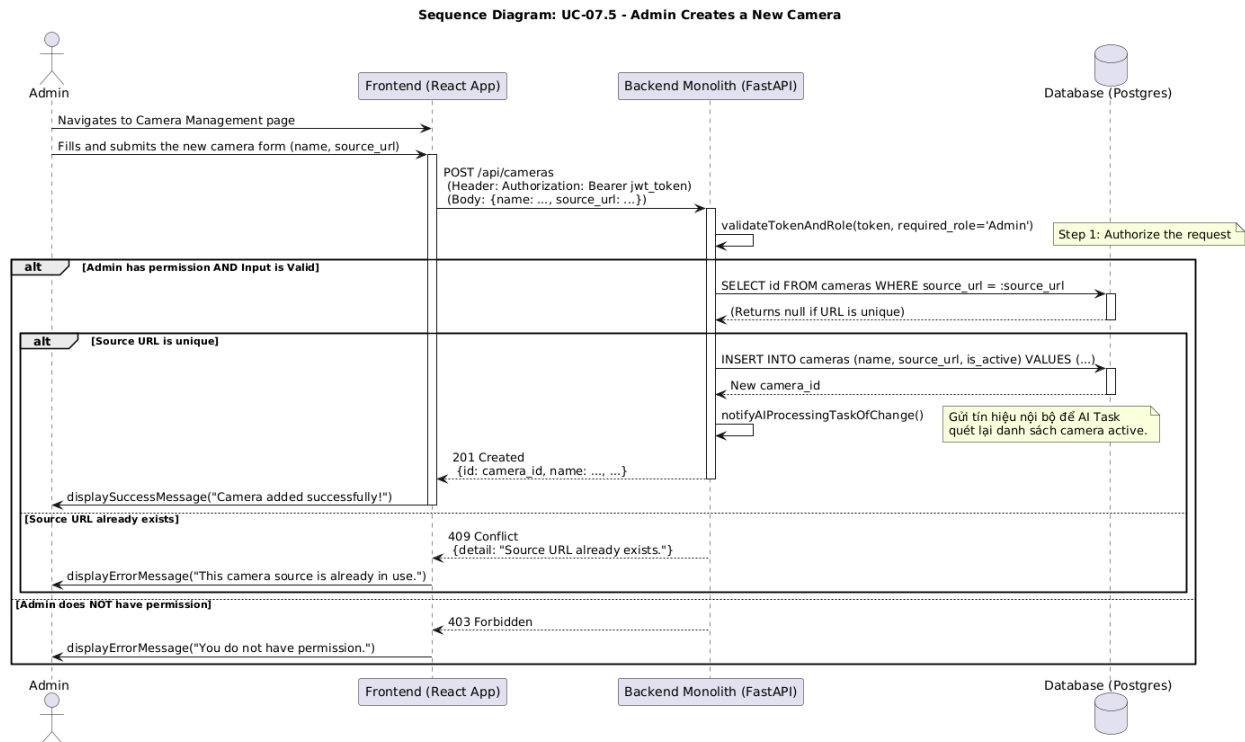
FE → Admin: displaySuccessMessage("Camera added successfully!")
deactivate FE
else Source URL already exists
BE → FE: 409 Conflict \n {detail: "Source URL already exists."}
deactivate BE

FE → Admin: displayErrorMessage("This camera source is already in use.")
deactivate FE
end
else Admin does NOT have permission
BE → FE: 403 Forbidden
deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")
deactivate FE
end

@enduml

```



1. **Admin** vào trang quản lý camera, điền form thêm camera mới và nhấn Lưu.
2. **Frontend** gửi request POST /api/cameras đến **Backend**, đính kèm JWT token.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** kiểm tra tính hợp lệ của dữ liệu (ví dụ source\_url không được trùng).
5. Nếu dữ liệu hợp lệ, **Backend** INSERT bản ghi mới vào bảng cameras trong **Database**.
6. **(Điểm mới)** Sau khi lưu thành công, **Backend** sẽ kích hoạt một cơ chế thông báo nội bộ (ví dụ: set một event flag, hoặc gửi message vào một queue nội bộ rất nhỏ) để AI Processing Task biết rằng cần phải cập nhật lại danh sách các camera đang chạy.
7. **Backend** trả về 201 Created để báo hiệu tạo thành công.
8. **Frontend** hiển thị thông báo thành công cho **Admin**.

## Use Case 7.6: Xem Danh sách Camera (Read Cameras)

@startuml

title Sequence Diagram: UC-07.6 - Admin Reads Camera List

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Navigates to Camera Management page

activate FE

FE → BE: GET /api/cameras \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission

' --- Step 2: Fetch the data ---

BE → DB: SELECT id, name, source\_url, is\_active FROM cameras

activate DB

DB → BE: List of all cameras

deactivate DB

BE → FE: 200 OK \n {cameras: [...]}

deactivate BE

FE → Admin: displayCamerasList(cameras)

deactivate FE

else Admin does NOT have permission

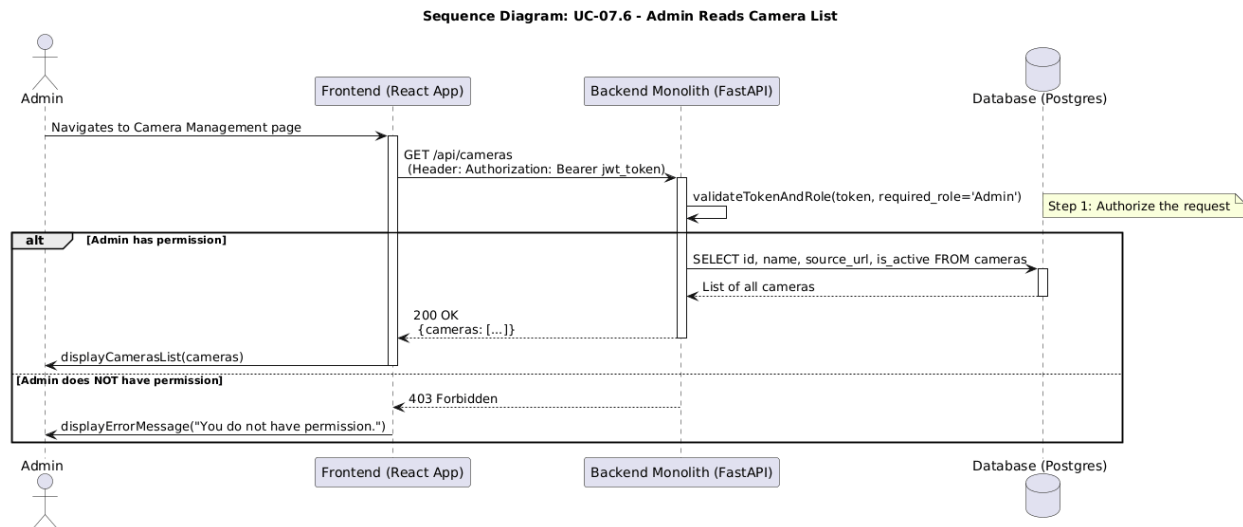
BE → FE: 403 Forbidden

deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")

deactivate FE  
end

@enduml



1. **Admin** truy cập vào trang "Quản lý Camera".
2. **Frontend** gửi request GET /api/cameras đến **Backend**, đính kèm JWT token.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** truy vấn **Database** để lấy toàn bộ danh sách camera.
5. **Backend** trả về danh sách này cho **Frontend** với status 200 OK.
6. **Frontend** nhận dữ liệu và hiển thị dưới dạng một bảng hoặc danh sách, cùng với các nút chức năng như Sửa, Xóa, Gán quyền...
7. Nếu không phải Admin, request sẽ bị từ chối với lỗi 403.

## Use Case 7.7: Cập nhật Thông tin Camera (Update a Camera)

@startuml

title Sequence Diagram: UC-07.7 - Admin Updates a Camera

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Clicks "Edit" on a camera in the list

Admin → FE: Modifies camera information (e.g., toggles 'is\_active') and submits

activate FE

FE → BE: PUT /api/cameras/{camera\_id} \n (Header: Authorization: Bearer jwt\_token) \n (Body: {name: ..., is\_active: false, ...})

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission AND Input is Valid

' --- Step 2: Update the camera record ---

BE → DB: UPDATE cameras SET name = ..., is\_active = ... WHERE id = :camera\_id

activate DB

DB → BE: (Update successful)

deactivate DB

' --- Step 3: Notify the AI processing module of the change ---

BE → BE: notifyAIProcessingTaskOfChange()

note right: Tín hiệu này sẽ khiến AI Task \nkhởi động lại/dừng tiến trình xử lý \ncho camera\_id tương ứng.

BE → FE: 200 OK \n {updated\_camera\_data}

deactivate BE

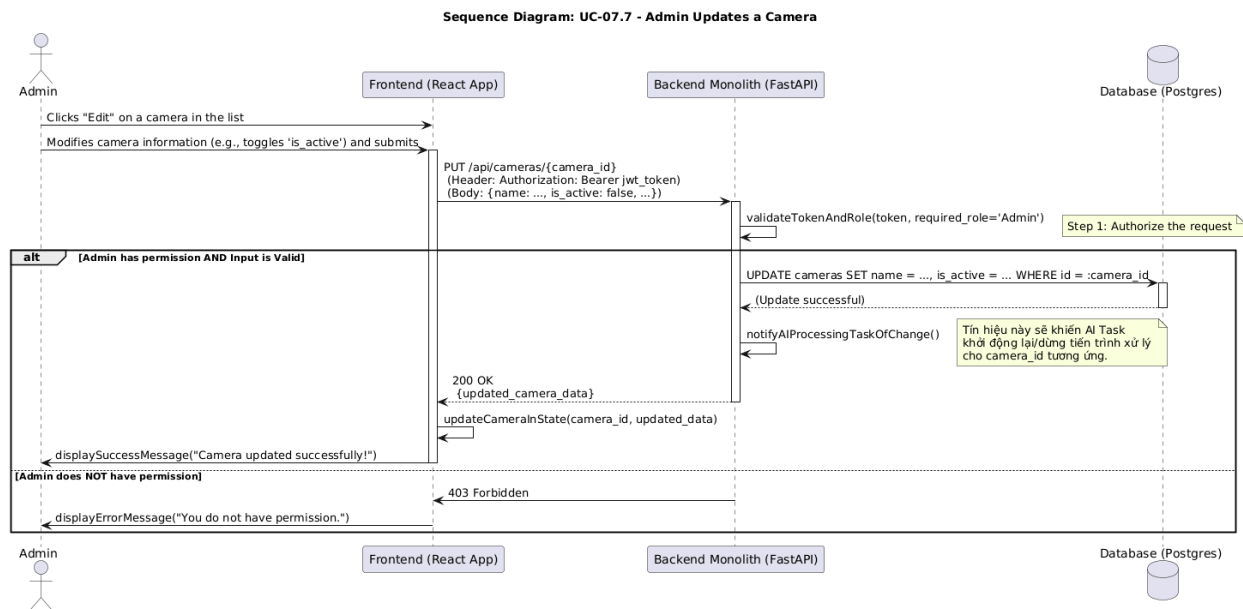
```

FE → FE: updateCameraInState(camera_id, updated_data)
FE → Admin: displaySuccessMessage("Camera updated successfully!")
deactivate FE
else Admin does NOT have permission
BE → FE: 403 Forbidden
deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")
deactivate FE
end

```

@enduml



1. **Admin** chọn một camera và nhấn "Sửa", sau đó thay đổi thông tin (ví dụ: bỏ tick ở ô "Đang hoạt động") và nhấn "Lưu".
2. **Frontend** gửi request PUT đến endpoint `/api/cameras/{camera_id}`.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin, **Backend** thực hiện lệnh UPDATE trong **Database** để cập nhật bản ghi camera.

5. **(Điểm mới)** Sau khi cập nhật DB, **Backend** gửi tín hiệu nội bộ để thông báo cho AI Processing Task về sự thay đổi. AI Task sẽ dựa vào tín hiệu này để:
  - Nếu is\_active chuyển từ true → false: Dừng tiến trình xử lý cho camera đó.
  - Nếu is\_active chuyển từ false → true: Khởi động tiến trình xử lý.
  - Nếu source\_url thay đổi: Dừng tiến trình cũ và khởi động tiến trình mới với URL mới.
6. **Backend** trả về 200 OK cho **Frontend**.
7. **Frontend** cập nhật lại giao diện và hiển thị thông báo thành công.

## Use Case 7.8: Xóa một Camera (Delete a Camera)

@startuml

title Sequence Diagram: UC-07.8 - Admin Deletes a Camera

actor Admin

participant "Frontend (React App)" as FE

participant "Backend Monolith (FastAPI)" as BE

database "Database (Postgres)" as DB

Admin → FE: Clicks "Delete" on a camera in the list

Admin → FE: Confirms the deletion

activate FE

FE → BE: DELETE /api/cameras/{camera\_id} \n (Header: Authorization: Bearer jwt\_token)

activate BE

BE → BE: validateTokenAndRole(token, required\_role='Admin')

note right: Step 1: Authorize the request

alt Admin has permission

' --- Step 2: Notify AI Task to stop processing ---

```

BE → BE: notifyAIProcessingTaskOfChange(camera_id_to_delete)
note right: Gửi tín hiệu để dừng ngay lập tức \ntiến trình xử lý cho camera n
ày.

' --- Step 3: Perform Hard Delete with Cascade ---
BE → DB: DELETE FROM cameras WHERE id = :camera_id
activate DB
note right of DB: ON DELETE CASCADE trigger: \n- Deletes related records i
n 'zones' \n- Deletes related records in 'violations' \n- Deletes related records
in 'police_camera_assignments'
DB → BE: (Deletion successful)
deactivate DB

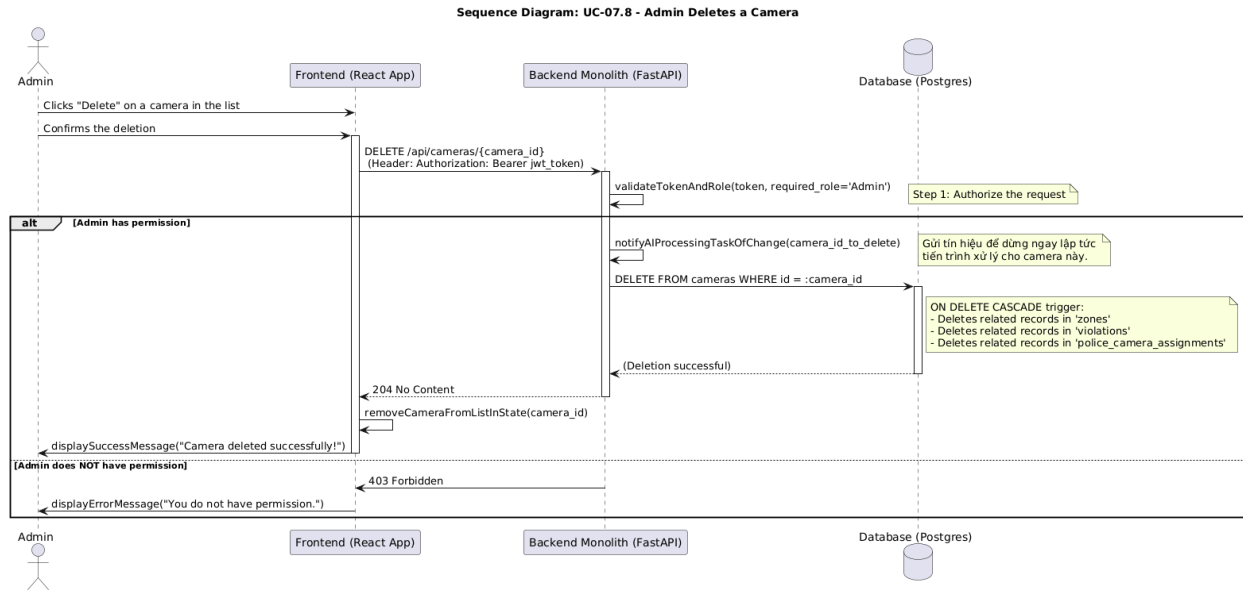
BE → FE: 204 No Content
deactivate BE

FE → FE: removeCameraFromListInState(camera_id)
FE → Admin: displaySuccessMessage("Camera deleted successfully!")
deactivate FE
else Admin does NOT have permission
BE → FE: 403 Forbidden
deactivate BE

FE → Admin: displayErrorMessage("You do not have permission.")
deactivate FE
end

@enduml

```



1. **Admin** nhấn nút "Xóa" trên một camera và xác nhận.
2. **Frontend** gửi request DELETE đến `/api/cameras/{camera_id}`.
3. **Backend** kiểm tra quyền Admin.
4. Nếu là Admin:
  - **Backend** đầu tiên sẽ gửi tín hiệu nội bộ để AI Processing Task dừng ngay lập tức tiến trình xử lý cho camera\_id sắp bị xóa.
  - Tiếp theo, **Backend** thực hiện lệnh DELETE FROM cameras... trong **Database**.
  - Nhờ có ràng buộc khóa ngoại ON DELETE CASCADE đã được định nghĩa ở tầng database, **Database** sẽ tự động xóa tất cả các bản ghi con liên quan trong các bảng zones, violations, police\_camera\_assignments. Điều này giúp Backend không cần phải viết logic xóa ở nhiều nơi.
5. **Backend** trả về status code 204 No Content.
6. **Frontend** xóa camera đó khỏi giao diện và hiển thị thông báo.