Chi tiết JSONS

Created	@December 15, 2025 2:54 PM
Tags	

1. Tổng quan hệ thống & Các phần chính tương tác

Hệ thống gồm 5 phần chính tương tác chặt chẽ:

1. **Frontend (React App)**: Giao diện người dùng (login, dashboard, config).
2. **Backend (FastAPI)**: Xử lý API, auth, CRUD, WebSocket push.
3. **Database (Postgres)**: Lưu users, sources (camera/video), rules, zones, violations.
4. **AI Pipeline + Rule Engine**: Xử lý video → detect/track → eval DSL → phát hiện vi phạm.
5. **Background Worker**: Task async xử lý stream/video từ sources.

Flow tổng quát:
User (Admin/Police) → Frontend → Backend (auth + API) → DB/AI Worker
AI Worker → Rule Engine → Violation → lưu DB + push WebSocket → Frontend dashboard

2. Tương tác chi tiết theo Use Cases (từ Usecases.pdf)

UC-01: Login

- **Tương tác** (theo Sequence Diagram chính xác):
    1. User nhập email/password → Frontend submit form.
    2. Frontend POST /api/token (body: {username: email, password}).
    3. Backend query DB: SELECT id, hashed_password, role FROM users WHERE email = ?
    4. Backend verify password (bcrypt) → nếu đúng: tạo JWT token (payload chứa id + role).

5. Backend trả {access_token, role}.
6. Frontend lưu token → redirect (Admin/Police → dashboard, Customer → my violations).

- **Phần liên quan**: Backend ↔ DB, Frontend lưu JWT cho các request sau.

UC-02: Real-time Dashboard (Police/Admin)

- **Tương tác**:
  1. User login → Frontend connect WebSocket /ws/dashboard (gửi token).
  2. Backend validate token + role → nếu Police: query DB lấy list source_id assigned (từ police_source_assignments).
  3. Backend accept connection → lưu client vào manager (filter by assigned sources nếu Police).
  4. Background Worker process active sources → AI detect → Rule Engine eval → nếu vi phạm: tạo violation record → lưu DB + push event qua WebSocket manager.
  5. WebSocket manager push processed frame (annotated) + violation alert đến client phù hợp.
  6. Frontend display live video + alert real-time.
- **Phần liên quan**: Frontend ↔ Backend (WebSocket), Backend ↔ Worker ↔ AI ↔ Rule Engine ↔ DB.

UC-03: Manage Configuration (Admin only)

- **Tương tác**:
  1. Admin vào trang config → vẽ zones (polygon) trên video source → Frontend gửi coordinates.
  2. Frontend POST/PUT /api/zones (JWT admin) → Backend validate role → lưu DB (zones table, coordinates JSONB).
  3. Admin viết DSL rule (text) → Frontend POST /api/rules → Backend parse ANTLR4 validate syntax → lưu DB (rules.dsl_content).
  4. Backend notify Worker reload rules/zones cho source đó.

- **Phần liên quan**: Frontend → Backend → DB → Worker (reload config).

UC-04: Assign source to police (Admin)

- **Tương tác**:
  1. Admin chọn police + sources → Frontend POST /api/assignments.
  2. Backend validate admin → update police_source_assignments table (delete old, insert new).
  3. Khi Police xem dashboard → backend filter source theo assignment.
- **Phần liên quan**: Frontend → Backend → DB.

UC-05 & UC-06: View/Filter Violations

- **Tương tác**:
  - Police: GET /api/violations?date=...&type=... → Backend query violations WHERE source_id IN (assigned sources).
  - Customer: GET /api/me/violations → Backend query WHERE detected_license_plate = user.license_plate.
  - Trả list violations + evidence_url.
- **Phần liên quan**: Frontend → Backend → DB.

3. Tương tác cốt lõi: AI Pipeline + Rule Engine + Violation Detection (từ Violation rules DSL)

Flow chính (core của hệ thống):

1. Admin add source (live camera hoặc upload video) → lưu DB.
2. Background Worker loop active sources:
   - Pull frame từ source_url hoặc file_path.
   - AI (YOLO) detect objects → ByteTrack track → calculate speed, direction, duration_in_zone.
   - Output JSON per frame (objects với type, bbox, speed, zone, helmet...).
3. Rule Engine load rules + zones của source từ DB.
4. Eval DSL trên JSON:

- Group 1: IF object.type == "car" AND object.current_zone == "motorbike_lane" THEN trigger.
- Group 2: IF speed < 5 AND duration_in_zone("no_parking") > 10s THEN trigger.

5. Nếu trigger → tạo Violation record (source_id, rule_id, timestamp, license_plate nếu OCR, metadata JSONB, evidence_url snapshot/clip).
6. Lưu DB → push alert qua WebSocket → dashboard hiển thị ngay.

---

1. UC-01: Log in to the System (Đăng nhập – Police/Admin/Customer)

Flow: Frontend gửi email/password → Backend verify → trả JWT token với role → Frontend redirect.

Input JSON (POST /api/token):

```
{
  "username": "admin@system.com",   // email
  "password": "admin123"            // plain text
}
```

Output JSON (200 OK):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.xxxx",
  "token_type": "bearer",
  "role": "admin"                   // dùng để frontend redirect
}
```

Bảng DB tương tác:

- **users** (đọc):
  - email (unique index)

- hashed_password (verify)
- role (trả về trong token)
- is_active (check nếu soft delete)

2. UC-02: View Real-time Monitoring Dashboard (Xem Dashboard Thời Gian Thực – Police/Admin)

Flow: Frontend connect WebSocket → Backend validate role + assigned sources → Worker push frame/violation alert.

Input (WebSocket connect – query param):

```
{
  "token": "JWT_token_here"
}
```

Output JSON Events (push qua WebSocket):

- **Processed Frame**:

  ```
  {
    "type": "processed_frame",
    "source_id": "uuid-source",
    "timestamp": "2025-12-15T10:00:00Z",
    "frame_base64": "..."
  }
  ```

- **New Violation Alert**:

  JSON

  ```
  {
    "type": "new_violation",
    "violation_id": "uuid-violation",
    "source_id": "uuid-source",
    "rule_name": "Đi không đúng làn đường",
    "timestamp": "2025-12-15T10:00:05Z",
  ```

```
  "object_class": "car",
  "zone_name": "motorbike_lane",
  "evidence_snapshot": "/static/evidence/snap_123.jpg",
  "evidence_clip": "/static/evidence/clip_123.mp4"
}
```

Bảng DB tương tác:

- **police_source_assignments** (đọc – nếu Police):
  - user_id, source_id (filter sources được phép xem)
- **sources** (đọc):
  - id, is_active, source_type (lấy active sources để process)
- **violations** (ghi khi có vi phạm mới):
  - id, source_id, rule_id, timestamp, detected_license_plate, evidence_url, metadata

3. UC-03: Manage System Configuration (Quản Lý Cấu Hình – Admin)

Flow: Admin tạo/sửa/xóa rules (DSL) và zones (vẽ polygon).

Input JSON (POST /api/rules):

```
{
  "name": "Đi không đúng làn đường",
  "dsl_content": "IF object.class_name IN (\"car\", \"bus\") AND object.current_
zone == \"motorbike_lane\" THEN TRIGGER_VIOLATION"
}
```

Output JSON (201 Created):

```
{
  "id": "uuid-rule",
  "name": "Đi không đúng làn đường",
  "dsl_content": "...",
  "is_active": true,
```

```
  "created_by_id": "uuid-admin"
}
```

Input JSON (POST /api/zones)

```
{
  "source_id": "uuid-source",
  "name": "motorbike_lane",
  "coordinates": [[100,100],[400,100],[400,500],[100,500]]
}
```

Output JSON (201 Created):

```
{
  "id": "uuid-zone",
  "source_id": "uuid-source",
  "name": "motorbike_lane",
  "coordinates": [[100,100],[400,100],[400,500],[100,500]]
}
```

Bảng DB tương tác:

- **rules** (CRUD):
  - id, name, dsl_content, is_active, created_by_id
- **zones** (CRUD):
  - id, source_id, name, coordinates (JSONB)

4. UC-04: Assign Camera to Police (Gán Source Cho Cảnh Sát – Admin)

Flow: Admin gán sources cho police.

Input JSON (POST /api/assignments):

```
{
  "user_id": "uuid-police",
```

```
  "source_ids": ["uuid-source1", "uuid-source2", "uuid-source3"]
}
```

Output JSON (200 OK):

```
{
  "message": "Assigned successfully",
  "assigned_count": 3
}
```

Bảng DB tương tác:

- **police_source_assignments** (xóa cũ + ghi mới):
  - user_id (PK), source_id (PK)

5. UC-05: View and Filter Violation Reports for Police

Flow: Police xem/filter violations từ sources được assign.

Input (GET /api/violations – query params):

```
{
  "start_date": "2025-12-01",
  "end_date": "2025-12-31",
  "rule_id": "uuid-rule",
  "source_id": "uuid-source",
  "page": 1,
  "page_size": 20
}
```

Output JSON (200 OK – array):

```
[
  {
    "id": "uuid-violation",
    "source_id": "uuid-source",
    "rule_name": "Đi ngược chiều",
```

```
    "timestamp": "2025-12-15T10:00:05Z",
    "detected_license_plate": "59A1-12345",
    "evidence_url": "/static/evidence/snap_123.jpg",
    "metadata": { "class_name": "motorbike", "speed_kmh": 40 }
  }
]
```

Bảng DB tương tác:

- **violations** (đọc + filter):
  - id, source_id, rule_id, timestamp, detected_license_plate, evidence_url, metadata
- **police_source_assignments** (đọc filter source_id hợp lệ)

6. UC-06: View and Filter Violation Reports for Customer

Flow: Customer chỉ xem violations của biển số mình.

Input (GET /api/me/violations – query params tương tự UC-05)

Output JSON: Giống UC-05, nhưng tự động filter theo detected_license_plate

Bảng DB tương tác:

- **users** (đọc): license_plate (từ JWT user)
- **violations** (đọc): detected_license_plate, các cột như trên

7. UC-07: Manage Infrastructure (Quản Lý Users & Sources – Admin)

Input JSON (POST /api/users – tạo user):

```
{
  "email": "police2@system.com",
  "password": "police123",
  "role": "police",
  "full_name": "Police Officer 2",
  "license_plate": null → optional
}
```

Output JSON (201):

```
{
  "id": "uuid-user",
  "email": "police@system.com",
  "role": "police",
  "full_name": "Police Officer",
  "is_active": true
}
```

Input JSON (POST /api/sources – thêm source):

```
{
  "name": "Ngã tư Phạm Văn Đồng",
  "source_type": "camera",
  "camera_url": "rtsp://admin:12345@192.168.1.100:554/stream1",
  "file_path": null,  // nếu uploaded_video thì fill path
  "duration": null
}
```

Bảng DB tương tác:

- **users** (CRUD): id, email, hashed_password, role, license_plate, is_active
- **sources** (CRUD): id, name, source_type, camera_url, file_path, duration, uploaded_by, is_active

Tương tác nội bộ (AI → Rule Engine → Violation)

Input từ AI Pipeline (per frame):

```
{
  "source_id": "uuid-source",
  "frame_timestamp": "2025-12-15T10:00:00Z",
  "objects": [
    {
      "track_id": 42,
      "class_name": "car",
```

```
      "class_id": 0,
      "bbox": [100, 100, 300, 300],
      "confidence": 0.96,
      "speed_kmh": 52.3,
      "direction_angle": 95.0,
      "current_zone": "motorbike_lane",
      "zone_duration_seconds": { "motorbike_lane": 8.5 },
      "attributes": {
        "has_helmet": null,
        "license_plate_text": "59A1-12345"
      }
    }
  ]
}
```

Output

```
{
  "violation_id": "uuid-violation",
  "source_id": "uuid-source",
  "rule_id": "uuid-rule",
  "timestamp": "2025-12-15T10:00:05Z",
  "detected_license_plate": "59A1-12345",
  "evidence_url": "/static/evidence/snapshot_123.jpg",
  "metadata": { "confidence": 0.95, "bbox": [100, 100, 300, 300], "rule_triggere
d": "Wrong lane" }
}
```