Verox System

Comprehensive System Documentation

Generated: November 05, 2025

Table of Contents

1. System Overview

2. High-Level Architecture

3. Frontend Architecture

4. Backend Architecture

5. Data Flow

6. Key Features

7. Database Architecture

8. Configuration System

9. API Endpoints

10. Deployment Architecture

11. Performance Characteristics

12. Security & Privacy

13. Use Cases

14. System Strengths

1. System Overview

Verox (Verolux1st) is an AI-powered security checkpoint system designed for real-time video analysis, body checking, gate security monitoring, and comprehensive analytics. The system combines advanced computer vision, AI detection, and modern web interfaces to provide a complete security monitoring solution.

2. High-Level Architecture

2.1 Three-Tier Architecture

The system follows a three-tier architecture: **Frontend Layer:** React + Vite + Zustand (State Management) on Port 5173 **Backend Layer:** FastAPI + Python + YOLOv8 + Computer Vision on Ports 8002 (Main), 8001 (Reports), 8003 (Semantic) **Data & Model Layer:** SQLite Databases + YOLOv8 Models + Video Sources The frontend communicates with the backend via HTTP/REST API and WebSocket connections for real-time updates.

3. Frontend Architecture

3.1 Technology Stack

• React 18.2.0 - Modern UI framework • Vite - Fast build tool and dev server • Zustand - Lightweight state management • Framer Motion - Animation library • React Konva - Canvas rendering • Recharts - Data visualization

3.2 Main Components & Pages

Dashboard: System overview, real-time statistics, health monitoring **Simple Inference:** Real-time object detection, gate configuration, live video stream with overlay, detection statistics (FPS, object count), adjustable gate areas **Advanced Body Checking:** Queue management, group detection, ticket-based system, Batch/Sequential examination modes **Gate Security:** 3-layer pipeline monitoring, FSM state visualization, check completion tracking **Analytics:** Traffic patterns, behavior analysis, performance metrics **Reports:** Incident reports, compliance reports, multi-language support **Semantic Search:** Natural language search for events and incidents

3.3 State Management (Zustand Stores)

• auth.js - JWT authentication, user roles (Admin/Viewer) • status.js - System health, model status • events.js - Event data management • alerts.js - Alert management • ui.js - UI state (current page, sidebar) • lang.js - Internationalization • theme.js - Theme settings

4. Backend Architecture

4.1 Main Server Files

advanced_body_checking.py (Port 8002): Main backend server with real-time video processing, gate security features, queue management, and authentication & security. **gate_sop_checker.py:** 3-layer pipeline for gate security: • Layer 1: Perception (Object detection, tracking, zone analysis, pose estimation) • Layer 2: Events (Micro-events logging, session management) • Layer 3: Decisions (FSM per person, explainable scoring, hysteresis logic)

4.2 Supporting Backend Modules

• tracking_system.py - ByteTrack multi-object tracking • zone_utils.py - Polygon operations, point-in-polygon, jitter filtering • pose_estimator.py - YOLOv8-pose integration, gesture detection • event_system.py - Event logging and session management • fsm_decision.py - Finite state machine decision engine • gate_database.py - SQLite storage for events, sessions, completions • analytics_system.py - Analytics and reporting (Port 8001) • reporting_system.py - Report generation with multi-language support • semantic_search.py - Semantic search backend (Port 8003) • incident_report_generator.py - Multi-language incident reports

5. Data Flow

5.1 Real-Time Detection Flow

1. Video Source (Webcam/File/RTSP) 2. Frame Capture (OpenCV) 3. YOLOv8 Detection → Person/object detection with confidence filtering 4. Multi-Object Tracking (ByteTrack) → Track assignment, IoU matching, Re-ID 5. Zone Analysis → Gate area detection, guard anchor detection, proximity calculation 6. Pose Estimation (Optional) → Hand-to-torso detection, reach gesture detection 7. Event Generation → Micro-events logged, session management 8. FSM Decision Engine → State transitions, scoring calculation, check completion 9. WebSocket Broadcast → Real-time updates to frontend 10. Database Storage → Events logged, sessions tracked, completions recorded

6. Key Features

6.1 Real-Time Object Detection: • YOLOv8 models (YOLOv8n, YOLOv8n-pose) • Multiple classes: person, car, truck, bus, bicycle, motorcycle • Confidence threshold: 0.3 • Max detections: 50 objects • FP16 optimization for performance **6.2 Gate Security System:** • Configurable gate areas (normalized polygons) • Guard anchor zones • Real-time monitoring • Body checking alerts • Check completion detection **6.3 Advanced Body Checking:** • Group detection (spatio-temporal clustering) • Ticket-based queue management • Batch and Sequential examination modes • Guard identification and tracking • SLA monitoring (Yellow/Red alerts) **6.4 Analytics & Reporting:** • Traffic flow analysis • Behavior pattern detection • PPE compliance monitoring • Anomaly detection • Multi-language incident reports (EN, ID, ZH) **6.5 Security Features:** • JWT authentication • Role-based access control • Rate limiting • Security headers (CORS, CSP, XSS protection) • Audit logging • PII scrubbing **6.6 Multi-Language Support:** • Incident reports in multiple languages • Template-based generation • PDF export

7. Database Architecture

7.1 SQLite Databases

gate_security.db: • Events (micro-events) • Sessions (check sessions) • Contact events (proximity interactions) • Pose events • Check completions • Snapshots • Anomalies • Performance metrics **verolux_analytics.db:** • Analytics data • Traffic patterns • Behavior metrics **verolux_enterprise.db:** • Enterprise-level data • Reporting data **verolux1st.db:** • Main application data

8. Configuration System

8.1 Gate Configuration

The system uses JSON-based configuration files for: • Zone definitions (polygons) • Timer thresholds • Proximity settings • Scoring parameters • Real-time configuration updates • Save/Load functionality Configuration includes: • Gate ID and zone definitions • Timer settings (dwell times, interaction windows) • Proximity detection parameters • Pose estimation settings • Scoring system parameters • Guard anchor logic

8.2 Example Configuration Structure

```
{ "gate_id": "A1", "zones": { "gate_area": "gate_A1_polygon.json",
"guard_anchor": "guard_anchor_A1_polygon.json" }, "timers": {
"person_min_dwell_s": 6.0, "guard_min_dwell_s": 3.0,
"interaction_min_overlap_s": 1.2 }, "scoring": { "base": 0.6,
"contact_bonus": 0.2, "pose_bonus": 0.15, "threshold": 0.9 } }
```

9. API Endpoints

9.1 Main Backend (Port 8002)

• GET /health - Health check • GET /internal/health - Authenticated health check • POST /auth/login - User authentication • GET /config/gate - Get gate configuration • POST /config/gate - Update gate configuration • GET /video/source - Get current video source • POST /video/source - Change video source • GET /counts - Get object counting statistics • POST /counts/reset - Reset counters • WebSocket /ws?token=... - Real-time detection stream

9.2 Gate Security Endpoints

• GET /gate/completions - List completed checks • GET /gate/session/{id} - Session details + timeline • GET /gate/stats - Performance statistics • GET /gate/config - Current configuration • POST /gate/reset - Reset checker state • WebSocket /ws/gate-check - Gate check stream

9.3 Analytics (Port 8001)

• GET /analytics/traffic-flow - Traffic analysis • GET /analytics/behavior-patterns - Behavior analysis • GET /analytics/ppe-compliance - PPE compliance • GET /analytics/anomalies - Anomaly detection

9.4 Reporting (Port 8001)

• GET /reports/incidents - Incident reports • POST /reports/export - Export reports • GET /reports/compliance - Compliance reports

9.5 Semantic Search (Port 8003)

• POST /search/query - Semantic search • GET /search/analytics - Search analytics

10. Deployment Architecture

10.1 Development Setup

• Backend: Python FastAPI on port 8002 • Frontend: Vite dev server on port 5173 • Models: Local YOLOv8 models • Databases: Local SQLite files

10.2 Production Options

• Docker containers (multiple Dockerfiles available) • Kubernetes deployment configs • GCP deployment configurations • Nginx reverse proxy • SSL/TLS support • Monitoring with Prometheus/Grafana

11. Performance Characteristics

• Processing Speed: ~5-20 FPS (configurable) • Latency: <200ms per frame • Memory Usage: ~500MB-2GB (depending on configuration) • CPU Usage: 15-30% (without GPU) • GPU Usage: 5-10% (with CUDA) • Scalability: Up to 10 simultaneous tracks per gate

12. Security & Privacy

• Local processing (no cloud dependency) • JWT-based authentication • Role-based access control • Rate limiting • Security headers (CSP, XSS protection) • Audit trail for compliance • Configurable snapshot retention • PII scrubbing capabilities • Database encryption (optional)

13. Use Cases

• Airport security checkpoints • Building entrance security • Event venue gate monitoring • Facility access control • Compliance monitoring and reporting • Real-time security analytics

14. System Strengths

1. Modular Architecture: Well-organized codebase with clear separation of concerns **2. Real-Time Processing:** WebSocket streaming for instant updates **3. Explainable AI:** FSM-based decision making with scoring system **4. Production-Ready:** Error handling, logging, monitoring, and scalability features **5. Configurable:** JSON-based configuration without code changes **6. Multi-Language Support:** Incident reports in multiple languages **7. Scalable:** Designed for multiple gates/cameras **8. Security-Focused:** Comprehensive security features and audit trails

Document generated on November 05, 2025 at 13:21:04