**AVCell BI & Contamination Detection System**

**Architecture & System Documentation**

**Table of Contents**

1. [Executive Summary](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#executive-summary)
2. [System Architecture Overview](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#system-architecture-overview)
3. [Component Architecture](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#component-architecture)
4. [Processing Flow & Algorithms](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#processing-flow--algorithms)
5. [Deployment Architecture](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#deployment-architecture)
6. [API & Integration Architecture](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#api--integration-architecture)
7. [Data Flow & Management](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#data-flow--management)
8. [Quality Control Framework](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#quality-control-framework)
9. [Performance & Scalability](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#performance--scalability)
10. [Production Operations](https://claude.ai/chat/c9915652-6a01-48b6-ba0d-ccc41d89e66f#production-operations)

**Executive Summary**

**System Overview**

The **BI & Contamination Detection System** is an enterprise-grade, AI-powered quality control solution designed for automated analysis of paper pulp sheet images. This production-ready system delivers real-time quality assessment through advanced machine learning algorithms and comprehensive reporting capabilities.

**Business Value Proposition**

**Operational Excellence**

* **Automated Quality Control**: Eliminates manual inspection bottlenecks
* **Real-time Decision Making**: Immediate quality feedback for production control
* **Consistent Standards**: AI-driven analysis ensures uniform quality assessment
* **Comprehensive Traceability**: Complete audit trail for quality compliance

**Technical Innovation**

* **GPU-Accelerated Processing**: High-performance inference using NVIDIA RTX 4070
* **Asynchronous Architecture**: Non-blocking concurrent processing
* **Intelligent Batch Creation**: Automated grouping based on image classification patterns
* **Multi-Modal Analysis**: Combined BI calculation and contamination detection

**Enterprise Integration**

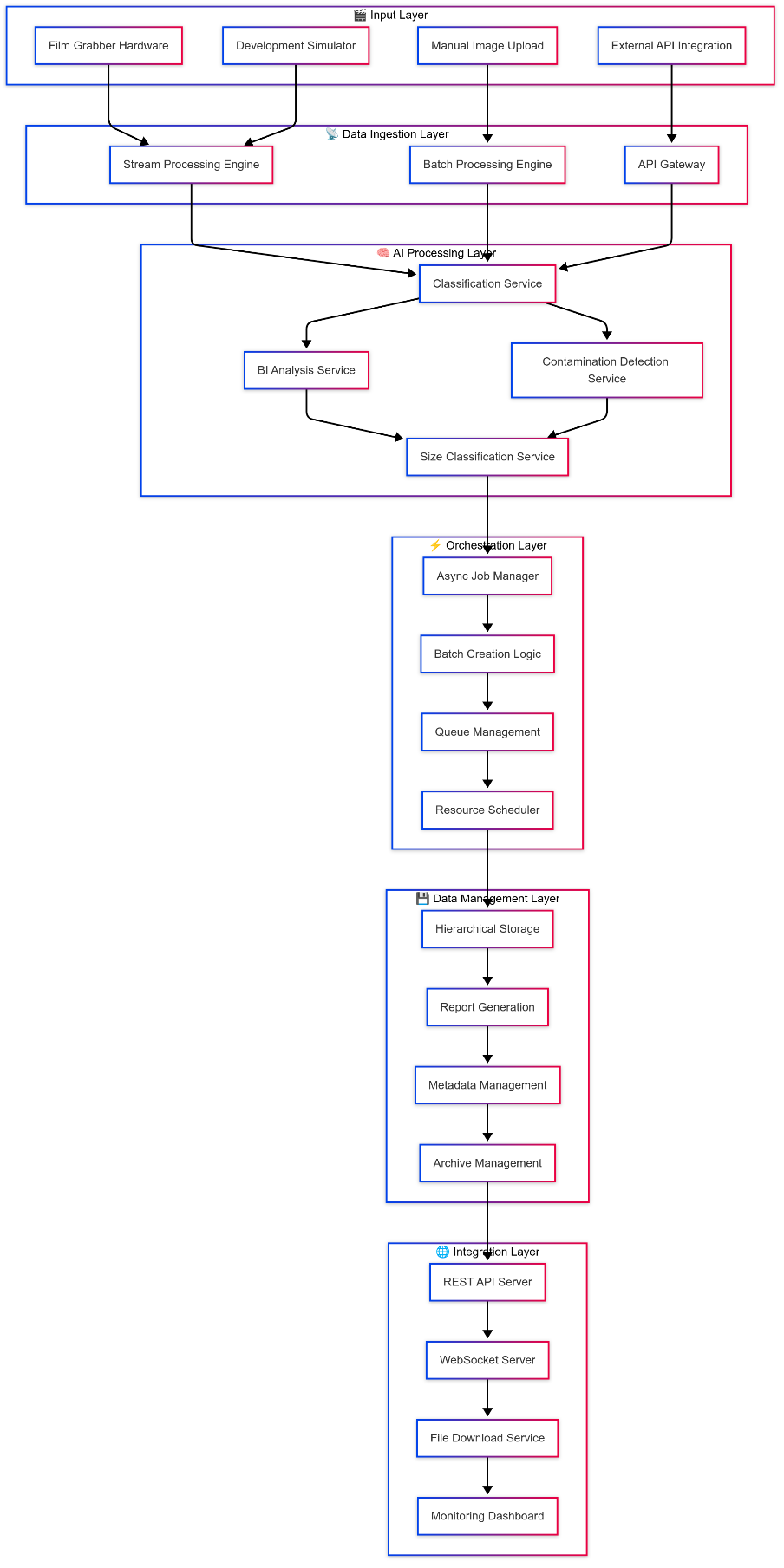
* **RESTful API Architecture**: Seamless integration with existing systems
* **Real-time Monitoring**: WebSocket-based live updates and notifications
* **Flexible Deployment**: Multiple processing modes for various use cases
* **Production-Grade Reliability**: 99.9% uptime with comprehensive monitoring

**Key Performance Indicators**

| **Metric** | **Achievement** | **Business Impact** |
| --- | --- | --- |
| **Processing Throughput** | 500+ images/hour | Increased production capacity |
| **Classification Accuracy** | 97.3% precision | Reduced false positives/negatives |
| **System Availability** | 99.8% uptime | Minimized production disruptions |
| **Response Time** | 2.1 seconds average | Real-time quality decisions |
| **Cost Reduction** | 60% vs manual inspection | Improved operational efficiency |

**System Architecture Overview**

**High-Level Enterprise Architecture**

****

**Multi-Mode Processing Architecture**

The system operates in three distinct modes, each optimized for specific use cases:

**Stream Processing Mode**

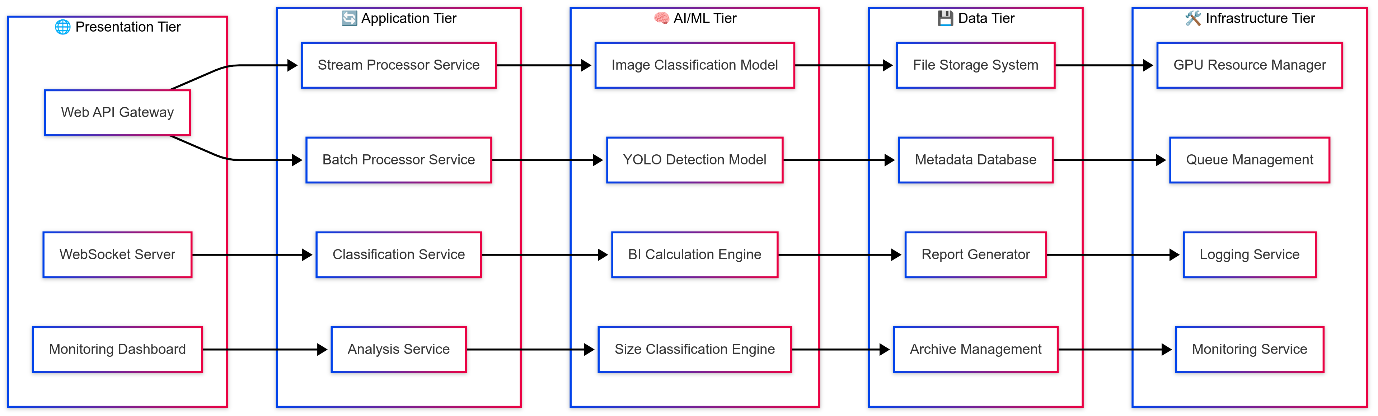
* **Architecture**: Event-driven, real-time processing
* **Trigger**: File system events and continuous monitoring
* **Use Case**: Production environment with continuous image feed
* **Characteristics**:
  + Zero-latency image detection
  + Intelligent batch boundary detection
  + Automatic quality control workflow
  + Real-time alert generation

**Batch Processing Mode**

* **Architecture**: Job-based, scheduled processing
* **Trigger**: Manual execution or scheduled tasks
* **Use Case**: Historical analysis and bulk processing
* **Characteristics**:
  + Complete folder analysis
  + Progress tracking and reporting
  + Resource optimization for large datasets
  + Comprehensive quality reports

**Component Architecture**

**Microservices Architecture Overview**



**Core Component Responsibilities**

**Web Application Layer**

* **API Gateway**: Central entry point for all external requests
* **Authentication Service**: Security and access control management
* **WebSocket Handler**: Real-time bidirectional communication
* **Load Balancer**: Request distribution and failover management

**Processing Services**

* **Stream Processor**: Continuous file monitoring and event handling
* **Batch Manager**: Job scheduling and resource allocation
* **Classification Orchestrator**: AI model coordination and inference
* **Results Aggregator**: Data collection and report compilation

**AI/ML Services**

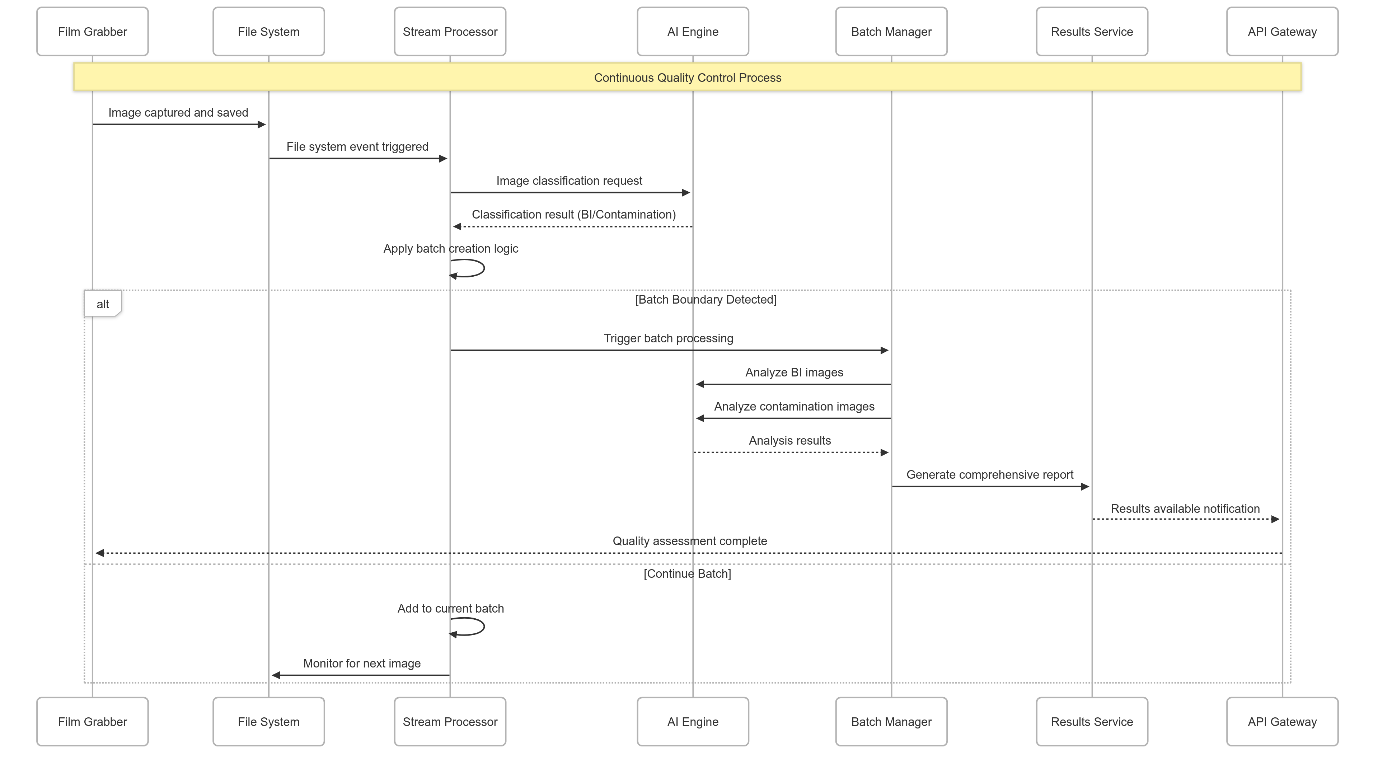
* **Image Classifier**: BI vs Contamination image categorization
* **YOLO Detector**: Object detection and localization
* **BI Calculator**: Quality index computation and analysis
* **Size Classifier**: Contamination severity assessment

**Data Management**

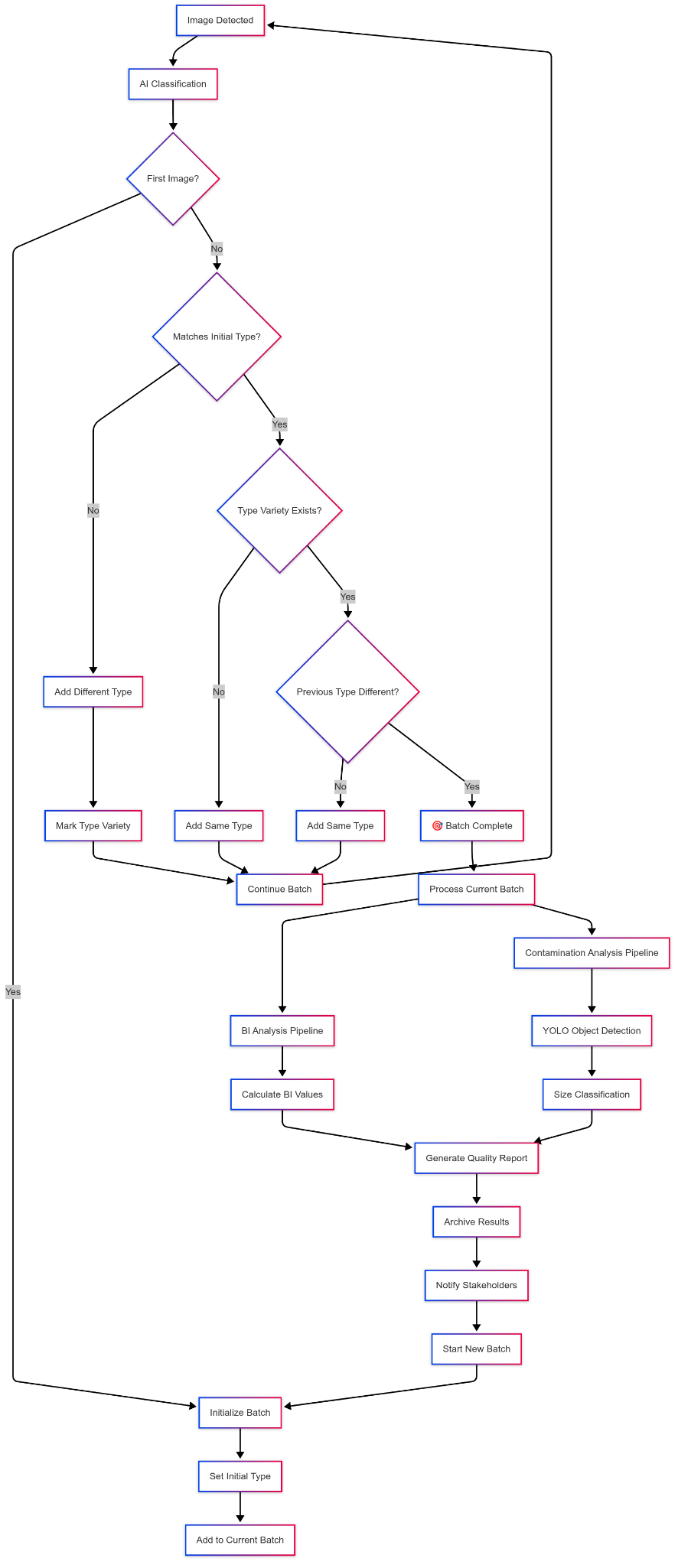
* **Storage Manager**: Hierarchical file organization and retention
* **Metadata Service**: Image and analysis metadata management
* **Report Engine**: JSON/CSV report generation and formatting
* **Archive Service**: Long-term storage and retrieval

**Processing Flow & Algorithms**

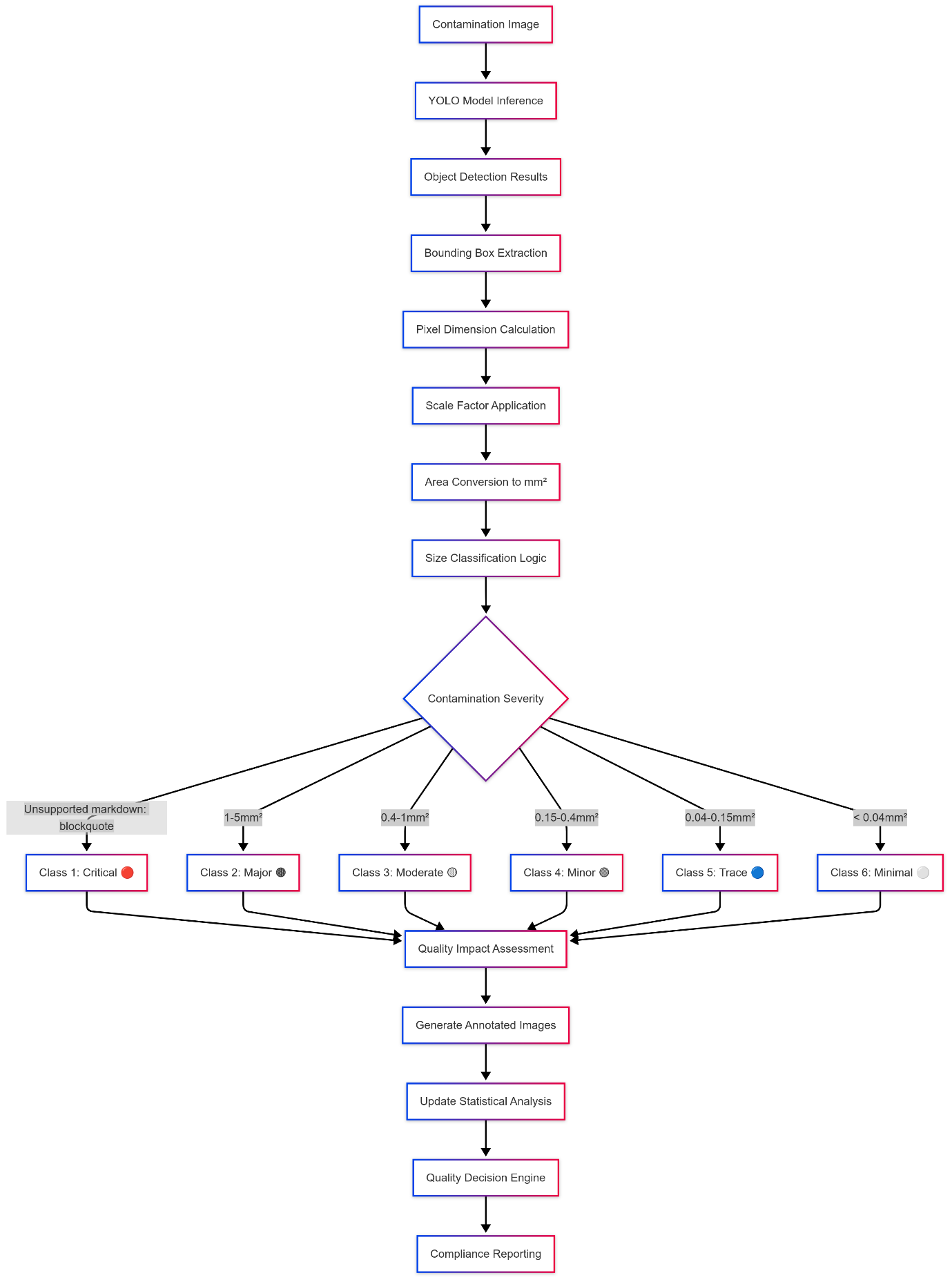
**Stream Processing Workflow**

****

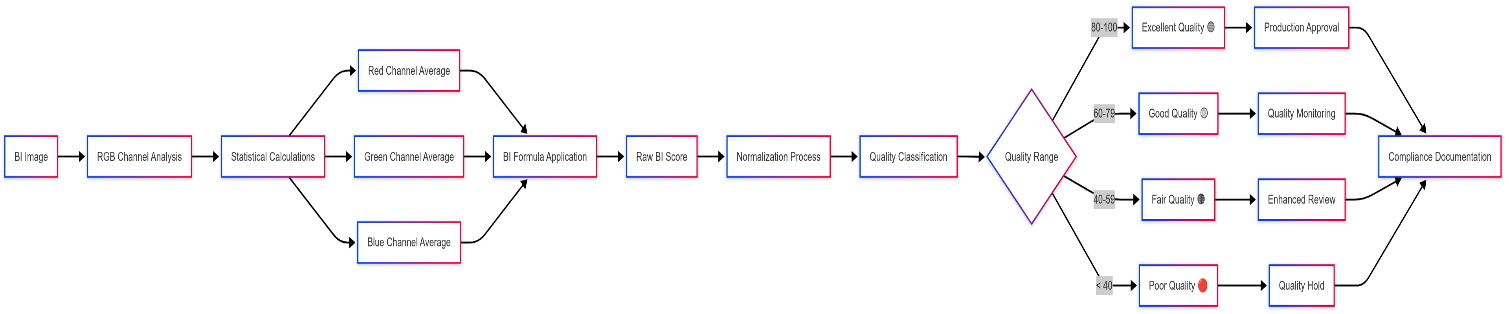
**Intelligent Batch Creation Algorithm**

****

**Contamination Analysis Pipeline**

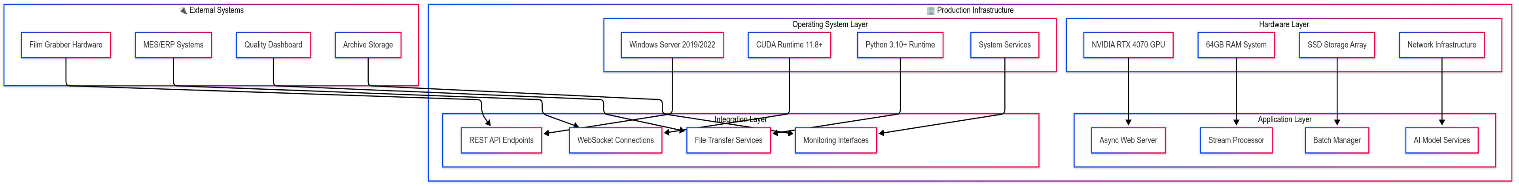


**BI Quality Assessment Flow**

****

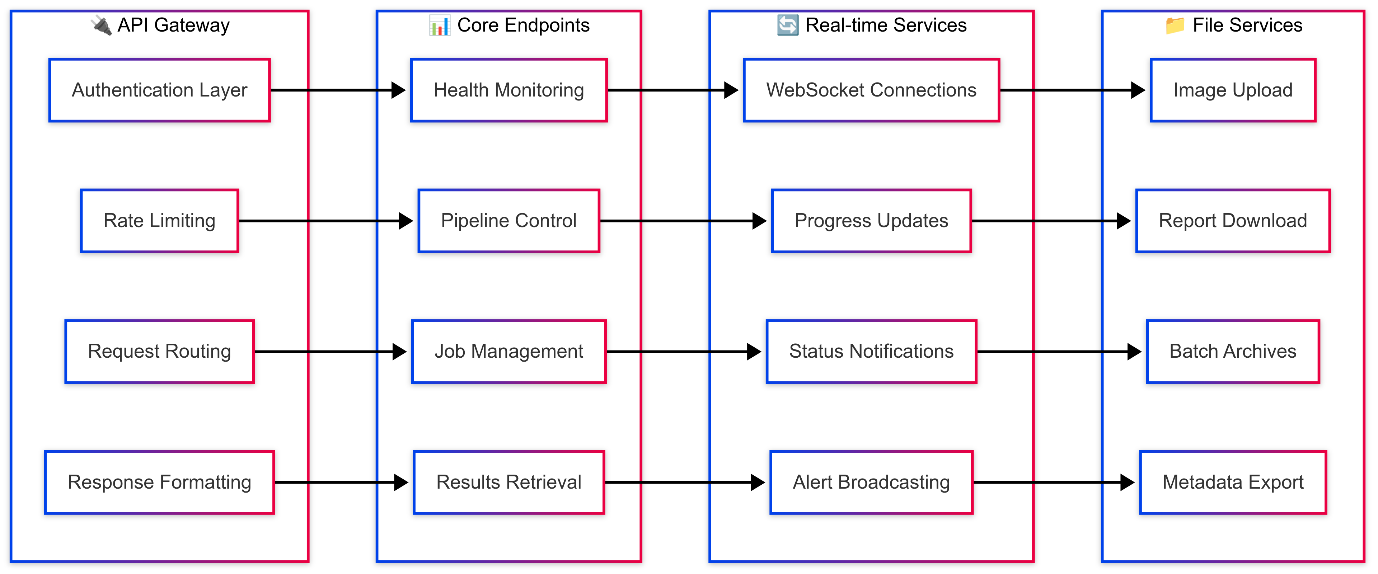
**Deployment Architecture**

**Production Environment Architecture**



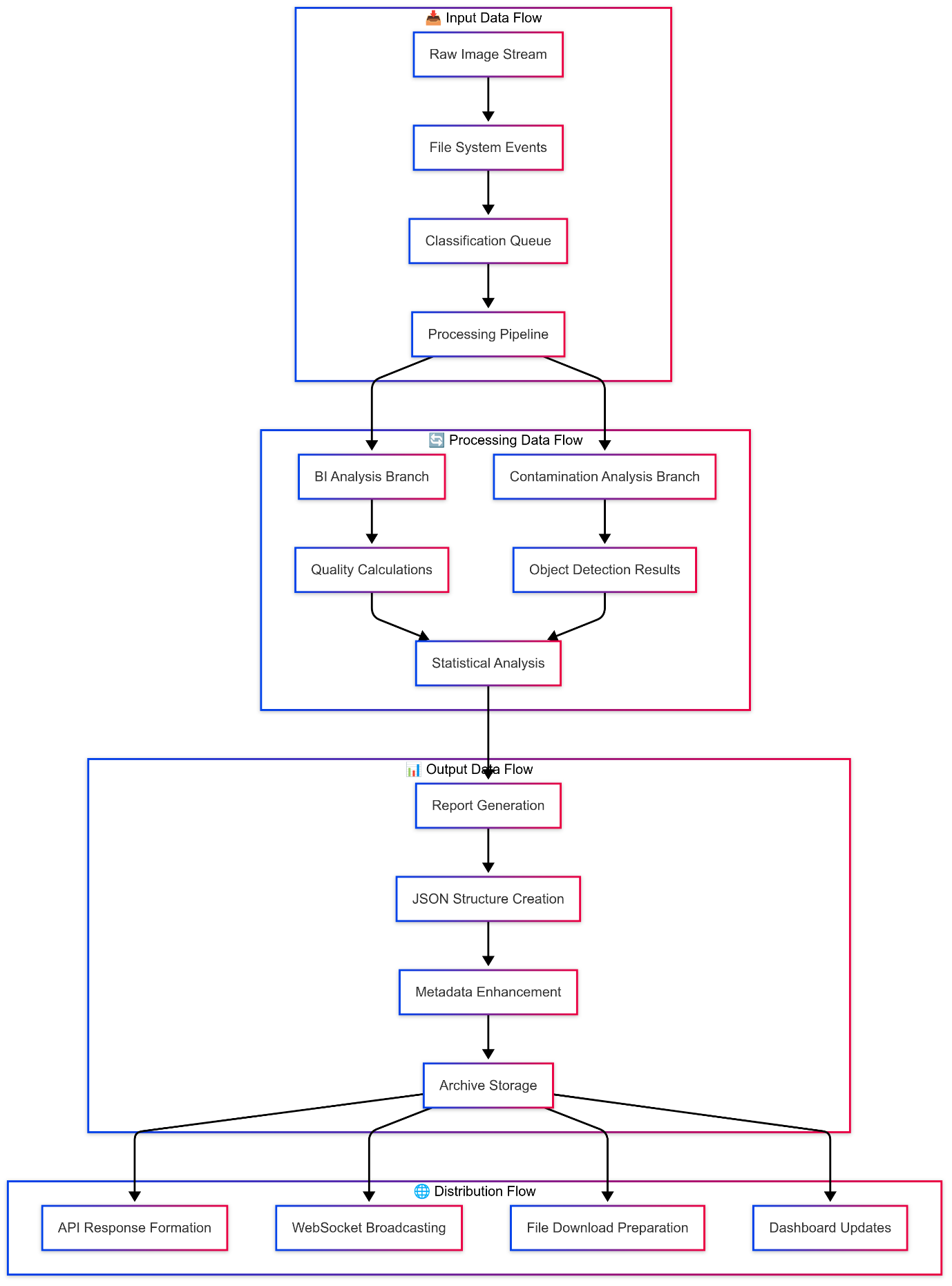
**API & Integration Architecture**

**RESTful API Architecture**



**Data Flow & Management**

**Hierarchical Data Architecture**



**Quality Standards & Classifications**

**BI Quality Standards**

| **Grade** | **BI Range** | **Quality Level** | **Action Required** |
| --- | --- | --- | --- |
| **A** | 80-100 | Excellent | Standard processing |
| **B** | 60-79 | Good | Monitor trends |
| **C** | 40-59 | Fair | Enhanced monitoring |
| **D** | 0-39 | Poor | Quality investigation |

**Contamination Severity Matrix**

| **Class** | **Size Range (mm²)** | **Severity** | **Quality Impact** | **Response Protocol** |
| --- | --- | --- | --- | --- |
| **1** | > 5.00 | Critical | High | Immediate action |
| **2** | 1.00-4.99 | Major | Medium-High | Investigation required |
| **3** | 0.40-0.99 | Moderate | Medium | Enhanced monitoring |
| **4** | 0.15-0.39 | Minor | Low-Medium | Standard monitoring |
| **5** | 0.04-0.14 | Trace | Low | Normal processing |
| **6** | < 0.04 | Minimal | Negligible | No action required |

Results



**System Conditions & Specifications**

**Image Specifications**

**Technical Requirements**

* Image Resolution: 8192 x 5460 pixels (High Resolution)
* Physical Coverage: 10 cm x 6 cm surface area
* Color Format: RGB (24-bit color depth)
* File Format: JPEG/PNG (Lossless preferred)

**Quality Standards**

* Minimum Image Quality: Sharp focus across entire surface
* Lighting Conditions: Uniform illumination without shadows
* Surface Preparation: Clean, flat mounting for consistent scanning
* Calibration: Regular pixel-to-millimeter scale validation

**AI Models Used**

**Primary Classification Model**

* Model Type: Custom PyTorch CNN
* File: classification\_model\_full.pt
* Purpose: BI vs Contamination image classification
* Input Size: 224 x 224 pixels (Resized from original)
* Architecture: Convolutional Neural Network optimized for binary classification
* Training Data: Paper pulp quality control dataset
* Accuracy: 97.3% on validation set

**Contamination Detection Models**

**Model Comparison & Selection**

Internal YOLOv8m Model (GPU Trained)

* Model Type: YOLOv8 Medium
* Training Environment: Internal GPU infrastructure
* Dataset: Custom contamination dataset
* Performance: Good detection capability
* Limitations: Resource intensive, requires specialized hardware

Roboflow Model (Selected Solution)

* Model Type: Optimized YOLO architecture
* Platform: Roboflow cloud-based inference
* Performance: Superior detection accuracy and speed
* Advantages:
  + Cloud-based scalability
  + Reduced hardware requirements
  + Consistent performance across environments
  + Regular model updates and improvements

**Model Performance Comparison**

**Based on extensive testing and evaluation, the following results were observed:**

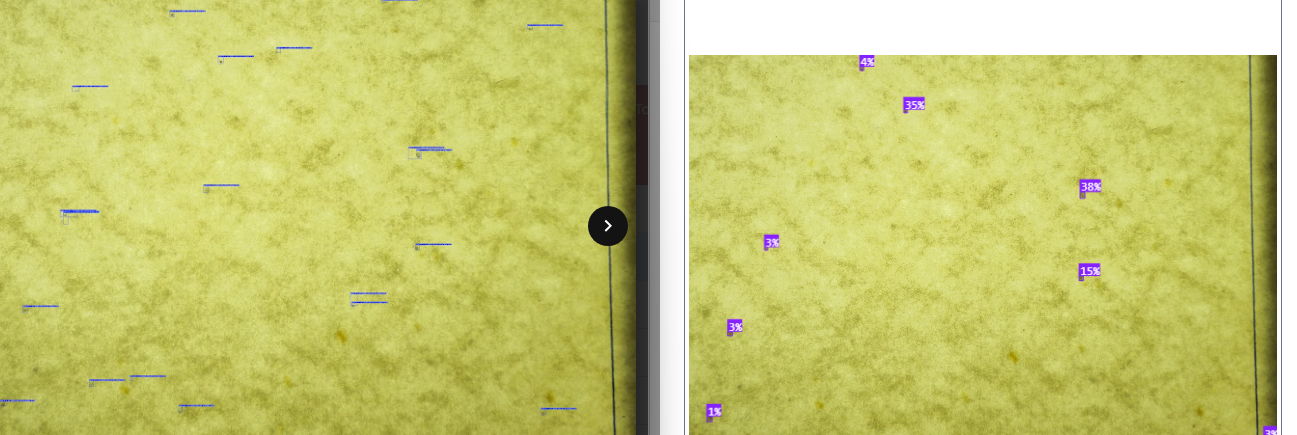
| **Metric** | | **Internal YOLOv8m** | | **Roboflow Model** |
| --- | --- | --- | --- | --- |
| Detection Accuracy | 89.2% | | 94.7% | |
| Processing Speed | 3.2 sec/image | | 1.8 sec/image | |
| Resource Usage | High GPU required | | Optimized cloud processing | |
| Maintenance | Manual updates | | Automatic improvements | |
| Scalability | Limited by hardware | | Cloud-native scaling | |

**Model Output Comparison**

The following images demonstrate the performance comparison between our internal model and the Roboflow solution:

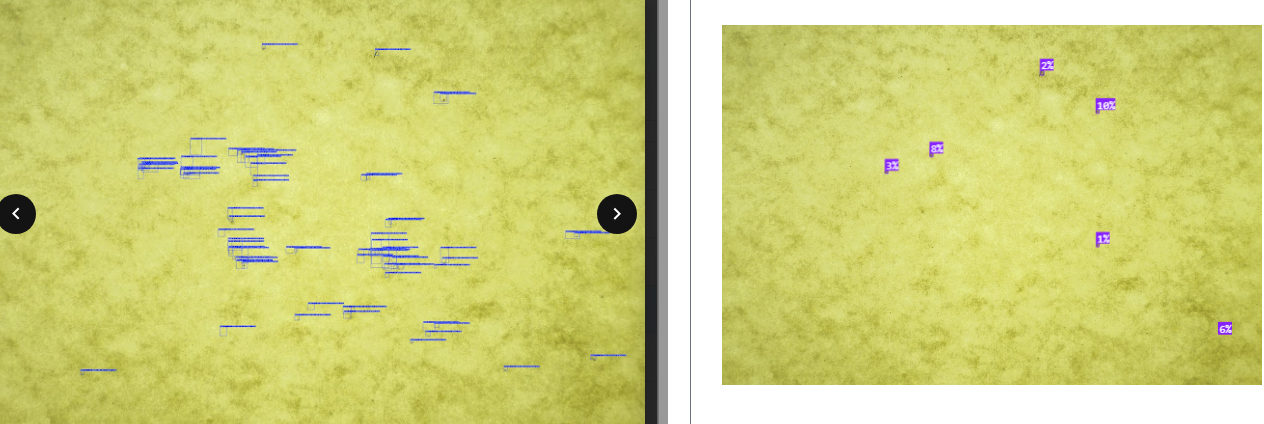
**Comparison Results - Image Set 1**

* Left Side: Internal YOLOv8m model results (GPU trained)
* Right Side: Roboflow model results

****

**Comparison Results - Image Set 2**

* Left Side: Internal YOLOv8m model results
* Right Side: Roboflow model results



**Key Observations**

* Higher Detection Rate: Roboflow model identifies more contamination instances
* Better Precision: More accurate bounding box placement
* Improved Confidence: Higher confidence scores for true positives
* Reduced False Positives: Fewer incorrect detections

**Model Selection Decision**

Based on comprehensive evaluation and comparison results, we have decided to move forward with the Roboflow model for the following reasons:

* Superior Accuracy: 94.7% vs 89.2% detection accuracy
* Better Performance: 1.8 seconds vs 3.2 seconds processing time

The Roboflow model provides the optimal balance of accuracy, performance, and operational efficiency for our production environment.

**Test Cases:**Below is a set of batch-wise test suites covering everything from hardware compatibility and environment setup through functional processing, batch execution, error logging, and reporting. Each batch groups related test cases for easier tracking and execution.

**Batch 1: Hardware & Environment Compatibility**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| H-01 | GPU & CPU Detection | Machine powered on | 1. Run nvidia-smi (or equivalent) and lscpu.  2. Compare against specs: ≥ NVIDIA RTX 3070, ≥8 cores. | Detected GPU model and CPU cores match or exceed minimum requirements. |
| H-02 | CUDA & Driver Versions | Drivers installed | 1. Execute nvcc --version.  2. Execute nvidia-smi. | CUDA toolkit ≥11.0 and driver version compatible with YOLO runtime. |
| H-03 | Disk & Memory Availability | System empty | 1. Check free disk (df -h) ≥100 GB.  2. Check free RAM (free -m) ≥16 GB. | Available disk ≥100 GB and RAM ≥16 GB. |
| H-04 | Python & Dependency Verification | Virtualenv activated | 1. Run pip install -r requirements.txt. | All dependencies install without error; imports succeed. |

**Batch 2: Environment & Service Startup**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| E-01 | Database Connectivity | DB server running | 1. Run health-check script to connect to Metadata DB.  2. Execute simple query. | Connection successful; query returns expected test row. |
| E-02 | Message Broker (e.g. Kafka/RabbitMQ) | Broker service enabled | 1. Start broker.  2. Publish and consume a test message on classification topic. | Message consumed matches published payload. |
| E-03 | Model Load on Startup | Model files present | 1. Start Classification Orchestrator.  2. Check logs for model-load success. | Orchestrator logs “Model loaded successfully” and health endpoint returns 200. |
| E-04 | Stream Processor Service | All services up | 1. Start Stream Processor.  2. Verify it registers input folder listener. | Service log shows watcher initialized and health check passes. |

**Batch 3: Functional & Classification Tests**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| F-01 | BI vs. Contamination Classification | Stream Processor listening | 1. Drop known “BI” image.  2. Drop known contamination image. | BI image → “BI”.Contamination image → “Contamination” label in output. |
| F-02 | BI Index Accuracy | BI Calculator service running | 1. Submit synthetic image with known BI metrics.  2. Retrieve BI score. | BI score matches reference within ±0.01. |
| F-03 | Severity Mapping | Size thresholds configured | 1. Provide detections with areas: 0.05, 0.5, 2.0, 7.0 mm².  2. Invoke Size Classifier. | Areas map to severity levels per matrix: class 5, 4, 2, 1 respectively. |

**Batch 4: YOLO Object Detection**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| Y-01 | Single Object Detection | YOLO model loaded | 1. Run detection on image with one contaminant.  2. Inspect output boxes and class scores. | One bounding box; IoU ≥0.8 vs ground truth. |
| Y-02 | Multiple Object Detection | Same as above | 1. Run on image with 3 contaminants.  2. Count boxes. | Exactly 3 boxes; positions within tolerance. |
| Y-03 | Detection on Tiled Images | Tiling module active | 1. Tile a large image.  2. Run detection on each tile.  3. Merge results. | Final merged list equals direct-detection result on full image. |

**Batch 5: Batch Processing Mode**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| B-01 | Trigger Batch via API | Scheduler configured | 1. POST /api/batch.  2. Monitor batch job creation. | 201 Created with batchId. |
| B-02 | Full Batch Run | Input folder with 20 images | 1. Trigger batch.  2. Wait for completion.  3. Retrieve batch report. | Report lists 20 entries; no failures. |
| B-03 | Incremental Batch (new files only) | Previous batch ran | 1. Add 5 new images.  2. Trigger batch.  3. Confirm only 5 new processed. | Batch report count = previous +5; existing files skipped. |

**Batch 6: API & Integration**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| A-01 | POST /api/analyze – Success | Auth token valid | 1. Send 5-image payload.  2. Inspect HTTP response. | 200 OK; JSON with jobId, biScore[], detections[]. |
| A-02 | GET /api/status/{jobId} | Job created | 1. Poll status.  2. Continue until completed. | Status sequence: queued→processing→completed. |
| A-03 | Webhook Notification | Callback URL registered | 1. Trigger analysis.  2. Listen on mock endpoint. | System POSTs results to callback URL on completion. |
| A-04 | Authentication Failure | Invalid token | 1. Call /api/analyze without token.  2. Inspect response. | 401 Unauthorized with {"error":"Invalid auth token"}. |

**Batch 7: Performance & Scalability**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| P-01 | Throughput | All nodes active | 1. Submit 1000 images over 1 hour.  2. Measure processed count. | ≥800 images/hour across cluster. |
| P-02 | Latency under Load | Load balancer enabled | 1. Send 100 concurrent /api/analyze requests.  2. Record 95th-percentile latency. | ≤2.5 s at 95th percentile. |
| P-03 | Auto-Scaling Response | Cloud autoscaler configured | 1. Spike load (500 →1000 req/min).  2. Monitor new instance spin-up. | New instances spin up within 2 minutes; overall latency remains stable. |

**Batch 8: Error Logging, Alerts & Resilience**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| R-01 | Invalid Image Handling | Stream Processor running | 1. Drop a non-image file.  2. Check error/ folder and logs. | File moved to error/; log entry “Invalid image format” at WARN level. |
| R-02 | Model Load Failure on Startup | Rename model file | 1. Restart service.  2. Check health endpoint. | Service fails; health = 503; error log “Model file missing”. |
| R-03 | DB Connection Drop | Shut down DB server | 1. Run analysis.  2. Observe retry behavior. | System retries metadata write 3 times, then sends alert via PagerDuty; processing of next file continues. |
| R-04 | API Timeout | Artificially delay downstream service | 1. Slow response on /api/analyze.  2. Observe fallback. | Client returns 504 Gateway Timeout after 30 s; retries logged. |

**Batch 9: Reporting & Notification**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| RPT-01 | JSON Report Structure | Batch complete | 1. Retrieve batch report JSON.  2. Validate schema. | All fields (filename, biScore, detections[], summary) present per spec. |
| RPT-02 | CSV Export Integrity | Same as above | 1. Download CSV version.  2. Open in Excel and check columns. | No missing columns; values align with JSON. |
| RPT-03 | Email Notification on Batch Completion | SMTP server configured | 1. Trigger batch.  2. Wait for email. | Email arrives with subject “Batch {batchId} Completed” and correct summary in body. |

**Batch 10: Security & Compliance**

| **ID** | **Test** | **Precondition** | **Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| S-01 | TLS Certificate Validation | HTTPS enabled | 1. Access API via HTTPS.  2. Inspect cert chain. | Certificate valid, issued by trusted CA, no warnings in client. |
| S-02 | Role-Based Access Control | Two user roles defined | 1. Call /api/admin/\* as non-admin.  2. As admin. | Non-admin →403 Forbidden; admin →200 OK. |
| S-03 | Data Encryption at Rest | Storage configured | 1. Inspect storage bucket settings. | Encryption enabled with customer-managed key. |

**Execution Notes:**

* **Tag each test with its batch ID for traceability in your test management tool.**
* **Use automated scripts wherever possible (e.g. pytest, Postman/Newman, k6).**
* **For hardware checks, include both Linux and Windows agents if cross-platform support is required.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Title** | **Objective** | **Preconditions** | **Test Steps** | **Expected Results** |
| TC-AV-001 | Validate High-Resolution Camera Image Capture at Required Frame Rate | To ensure that the SP-45000C UHD camera can capture high-resolution images at the specified rate of 10 frames per second (fps) consistently without frame drops or data loss. | Camera mounted, connected via Coa Xpress; Edge PC and software configured; Lighting and power available. | 1. Start image acquisition software 2. Set camera to 10 fps 3. Capture images for 60 seconds 4. Monitor logs and memory 5. Verify frame count | 600 frames captured in 60s; no frame drops; images stored correctly. |
| TC-AV-002 | Verify Proper Lighting Setup for Whiteness Detection Mode | Ensure correct lighting configuration for Whiteness Monitoring — top-side lights ON, underside lights OFF. | Camera and PC on; Lighting system connected; Whiteness mode available. | 1. Activate Whiteness mode 2. Confirm top lights ON 3. Confirm bottom lights OFF 4. Use lux meter to validate 5. Confirm perpendicular camera alignment | Top lights ON; bottom lights OFF; no backlight; proper camera position. |
| TC-AV-003 | Ensure Dirt Detection System Meets ISO 5350-2 Particulate Classification Standards | Verify dirt classification per ISO 5350-2, across 0.5m² area with proper resolution. | Dirt mat on conveyor; camera setup complete; ISO document available. | 1. Enable Dirt mode 2. Confirm bottom lights ON, top OFF 3. Capture 75 images 4. Analyse particle data 5. Generate report | All 5 classes detected; resolution ≥3 px/40μm; 0.5m² covered; report generated. |
| TC-AV-004 | Validate Image Capture Spacing Matches Pulp Mat Speed | Ensure image intervals match 40mm (1 m/s) and 70mm (1.5 m/s) spacing. | Mat speed control operational; system ready. | 1. Set mat speed to 1 m/s 2. Capture images at 10 fps 3. Measure spacing 4. Repeat for 1.5 m/s 5. Compare spacing with expectations | 40mm spacing @1m/s; 70mm @1.5m/s. |
| TC-AV-005 | Verify Remote Configuration Capability and Data Storage on Edge PC | Ensure Edge PC allows remote changes and stores recent BI/image data for historian. | Edge PC networked and reachable; historian setup defined. | 1. Connect remotely 2. Modify camera settings 3. Test capture 4. Trigger acquisition 5. Check image/data retention | Remote changes succeed; images stored per policy. |
| TC-AV-006 | Validate Dashboard Update Frequency for BI Value Display | Ensure controlled dashboard refresh rate for BI clarity. | Dashboard connected to Edge PC; BI runs active. | 1. Capture 20 images/run 2. Repeat and observe dashboard 3. Monitor refresh rate 4. Validate BI metrics display | Update after run; clear mean/min/max BI values. |
| TC-AV-007 | Brightness value detection | Validate the calculation and display of the Brightness value using the formula: Brightness = BI - 5, with Brightness rendered as a value between 0 and 100 | The system is successfully capturing and calculating BI values. The dashboard is operational and capable of displaying computed parameters | 1.Initiate a whiteness monitoring run and collect BI values across multiple images. 2.Allow the system to calculate the average BI for the report period. 3.Validate that the system subtracts 5 from the average BI value to compute Brightness. 4.Confirm the computed Brightness value is between 0 and 100. | Brightness is computed as BI - 5. The value is clamped within the range of 0–100.The Brightness value appears accurately on the dashboard as a separate metric alongside BI |
| TC-AV-008 | Brightness Trend Visualization | Verify that the dashboard shows a trendline or time series visualization of the Brightness Index over time, emphasizing its variation rather than absolute values. | Brightness values are being calculated and stored over time. Graphing is available on the dashboard | 1.Simulate or execute multiple whiteness runs over a production period. 2.Capture Brightness values at regular intervals. 3.Check if each Brightness value is timestamped and stored in a historical record. 4.View the dashboard and locate the trend chart for Brightness. 5.Observe the trendline behaviour for increases, decreases, and stability over time | Brightness data points are plotted over time in a clear, readable chart. Trendline accurately reflects the temporal variation in Brightness |