**TEAM NAME**

VigilEye AI

**Team members:**

**1)G Abhinav Reddy (**[**abhinavreddyg08@gmail.com**](mailto:abhinavreddyg08@gmail.com)**)**

**2)E Anish**

**(anishedaaegon@gmail.com)**

**VISUAL DIFFERENCE ENGINE**

**1)Problem Statement:**

* Manual visual inspection in manufacturing, construction, and brand compliance is slow, error prone, and fatiguing for workers.
* Existing systems struggle with real world variations like lighting changes, camera angle shifts, and subtle texture differences; leading to false alarms or missed defects.
* Pixel-based comparison tools lack semantic understanding: they can’t distinguish between harmless noise for example shadows and meaningful changes like cracks, corrosion, missing parts.
* This results in safety risks, unnecessary downtime, costly recalls, and overburdened inspectors.
* There’s a need for a robust, general-purpose visual difference engine that detects, localizes, and classifies meaningful changes across time-series images understanding *what* changed and *why it matters*.
* This type of system would augment human expertise, improve decision making, and enable proactive, reliable monitoring across industries.

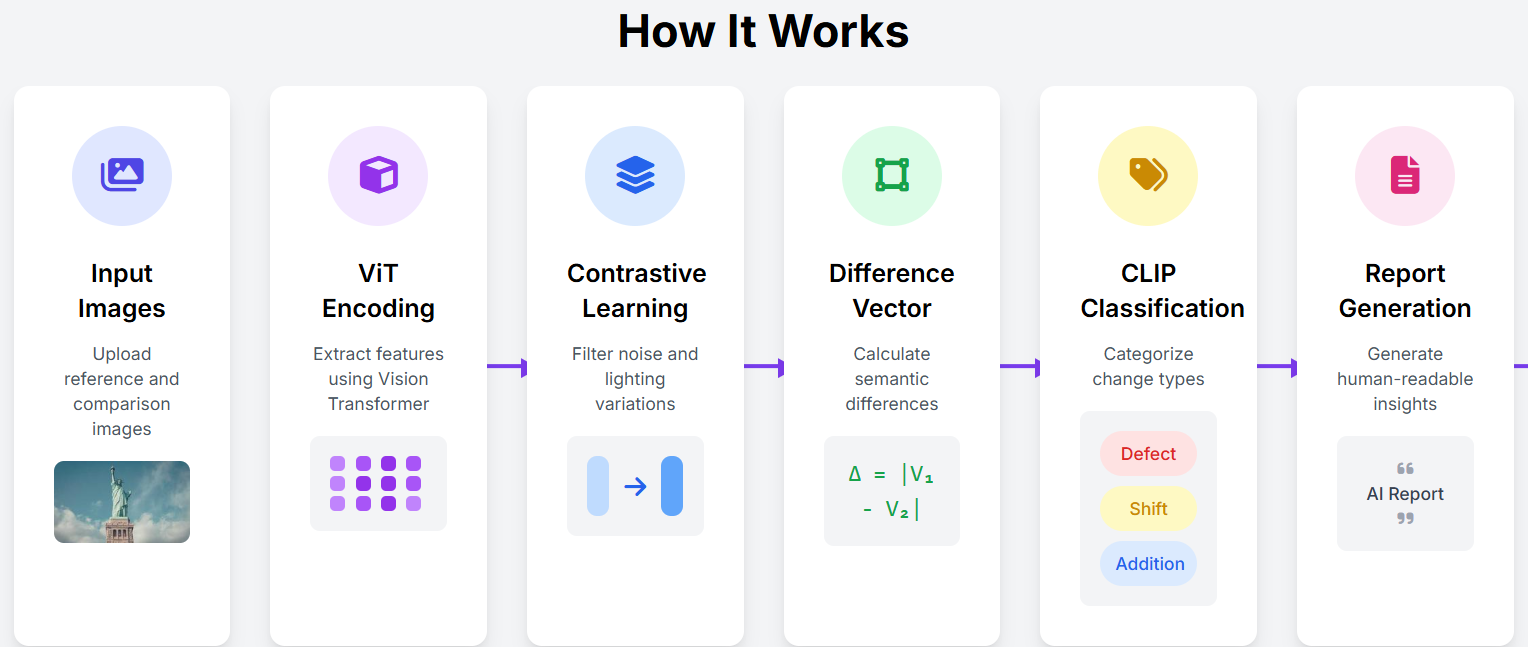
**2)Proposed solution:**

VigilEye AI is a context aware vision language difference engine that performs semantic level change detection between time series or sequential images. It integrates vision transformers, contrastive representation learning, and multimodal reasoning to identify, classify, and explain meaningful visual differences under real world variability.

**Architectural Overview:**

VigilEye AI consists of five primary modules working in sequence:

1. **Feature Extraction using Vision Transformer (ViT):**  
   Both input images are passed through a pre-trained ViT backbone (e.g., ViT-B/16 or Swin Transformer).  
   The model encodes global contextual features and fine-grained structural patterns into dense embeddings.  
   This enables VigilEye to compare images in a semantic latent space rather than pixel space.
2. **Contrastive Learning Module (SimCLR Framework):**  
   To ensure robustness against illumination changes, minor shifts, and camera variations, VigilEye is pretrained using a contrastive self supervised objective.  
   Augmented views of the same scene are optimized to produce similar embeddings, while distinct images produce dissimilar ones.  
   This stage builds invariance to irrelevant noise and improves domain generalization.
3. **Latent Difference Computation:**  
   Encoded embeddings from the two images are aligned and subtracted in feature space to obtain a difference vector (ΔF).  
   ΔF represents high-level semantic variation, filtering out lighting or geometric noise that affects raw pixels.
4. **Semantic Change Classification using CLIP:**  
   The ΔF vector is passed into a fine-tuned **CLIP model**, which maps visual changes to a multimodal (associates image with text) embedding space.  
   This allows VigilEye to classify changes into interpretable categories such as crack formation, object removal, surface deformation, or logo displacement.  
   CLIP’s language grounding enables consistent labeling without requiring large domain-specific datasets.
5. **Vision-Language Report Generation (BLIP / MiniGPT):**  
   A lightweight vision language transformer generates textual summaries of detected changes.  
   The model uses visual embeddings and class tokens to produce concise explanations.



**Key Technical Differentiators**

* **Semantic-Level Comparison:** Operates in latent feature space using ViT embeddings, avoiding pixel level false positives.
* **Contrastive Robustness:** Learns illumination and viewpoint invariance through self-supervised contrastive pretraining.
* **CLIP-Enhanced Classification:** Aligns visual differences with natural language labels for interpretable change reasoning.
* **Vision-Language Summarization:** Converts detection outputs into human-readable, context-aware inspection reports.
* **Scalable, Multi-Domain Design:** Single architecture adaptable to manufacturing, construction, brand audits, and environmental monitoring with minimal fine-tuning.

**Output and Visualization**

The system outputs:

* A semantic change map highlighting affected regions.
* A change classification report specifying category and severity.
* A natural language summary for explainable inspection.

VigilEye AI can be deployed via a FastAPI based backend, offering APIs for image comparison and report generation. The frontend dashboard visualizes annotated images and temporal heatmaps using React.js and Plotly.js.  
Optimized models can be exported to **ONNX or TensorRT** for real time inference on edge devices or embedded inspection systems.

**3) Functionality:**

1. Intelligent Change Detection:

VigilEye compares two or more images of the same scene and detects meaningful differences at both structural and semantic levels. It identifies physical alterations such as cracks, dents, object shifts, or brand element mismatches, while suppressing irrelevant variations from lighting, angle, or shadows. This ensures high precision and low false alarm rates in real world conditions.

2. Multi-Level Change Classification

Once a difference is detected, the system classifies it into contextual categories such as addition, removal, deformation, displacement, or surface defect. Each detected region is assigned a change type, confidence score, and severity index, enabling automated decision-making and prioritization of critical issues.

3. Robust Noise Filtering:

Through pretraining on contrastive objectives, VigilEye learns to distinguish environmental noise (glare, blur, illumination shifts) from genuine structural change. This makes it highly robust for dynamic environments like factory floors, outdoor sites, or drone based monitoring systems where traditional methods fail.

4. Spatial Heatmap Generation:

The system produces an interpretable heatmap that visually highlights areas of significant change. The intensity of each region corresponds to the degree of semantic difference, allowing users to instantly locate and assess anomalies without inspecting full image sequences manually.

5. Vision-Language Insight Reporting:

VigilEye translates its visual findings into natural-language summaries, providing human-interpretable inspection reports. This bridges AI perception and human communication, ensuring explainable, audit friendly outputs.

6. Temporal and Multi-View Handling:

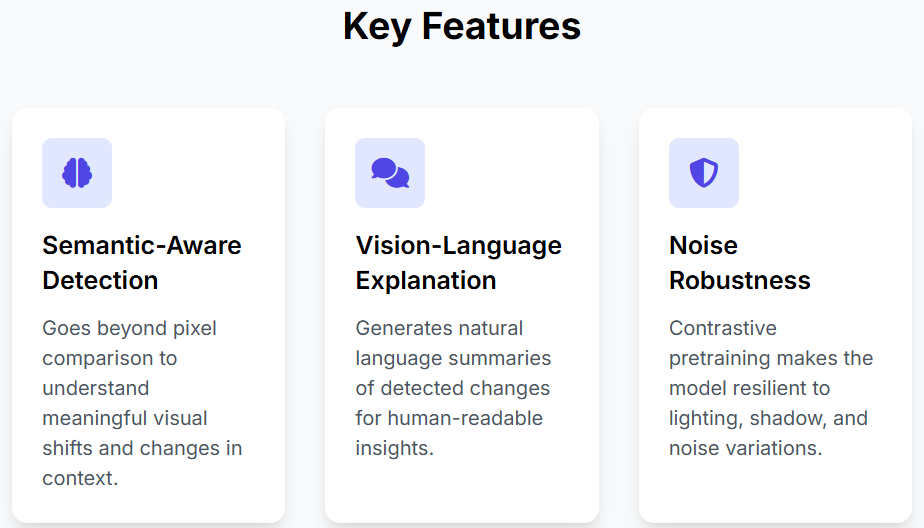
The engine supports multi frame and cross angle comparisons, enabling applications in timelapse inspection, drone surveys, or surveillance video analysis. It can track how an object or surface evolves over time, making it suitable for long term degradation or compliance monitoring.

7. Scalable Integration and Outputs:

VigilEye AI can be deployed as a backend inference API or integrated into an interactive dashboard. Its outputs include:

* Annotated image overlays with color coded difference regions.
* Machine-readable metadata (JSON/CSV) for automated systems.
* Human-readable inspection summaries for engineers or supervisors.

These outputs allow seamless integration into industrial pipelines, brand audit dashboards, or environmental monitoring systems.

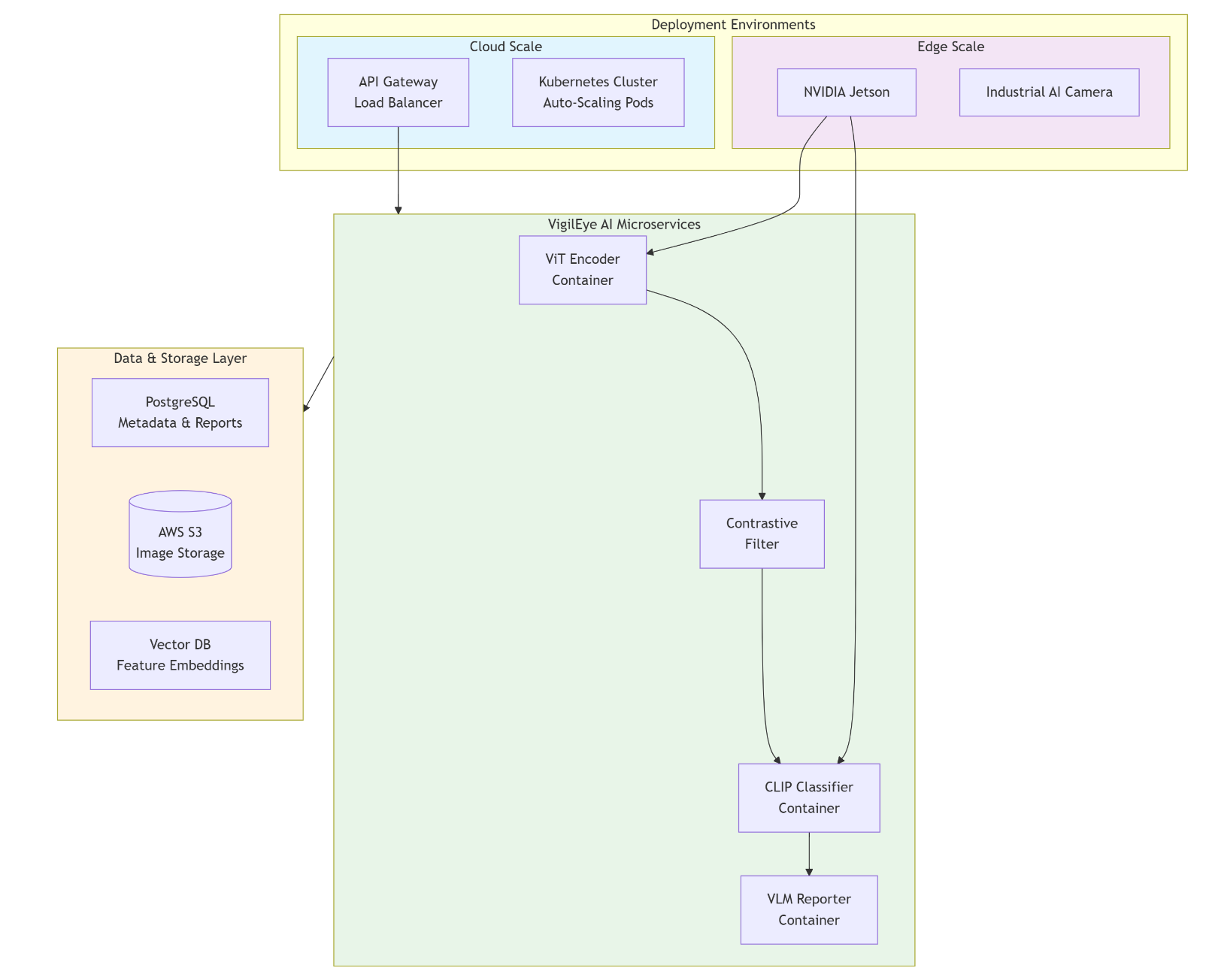


**Scalability**

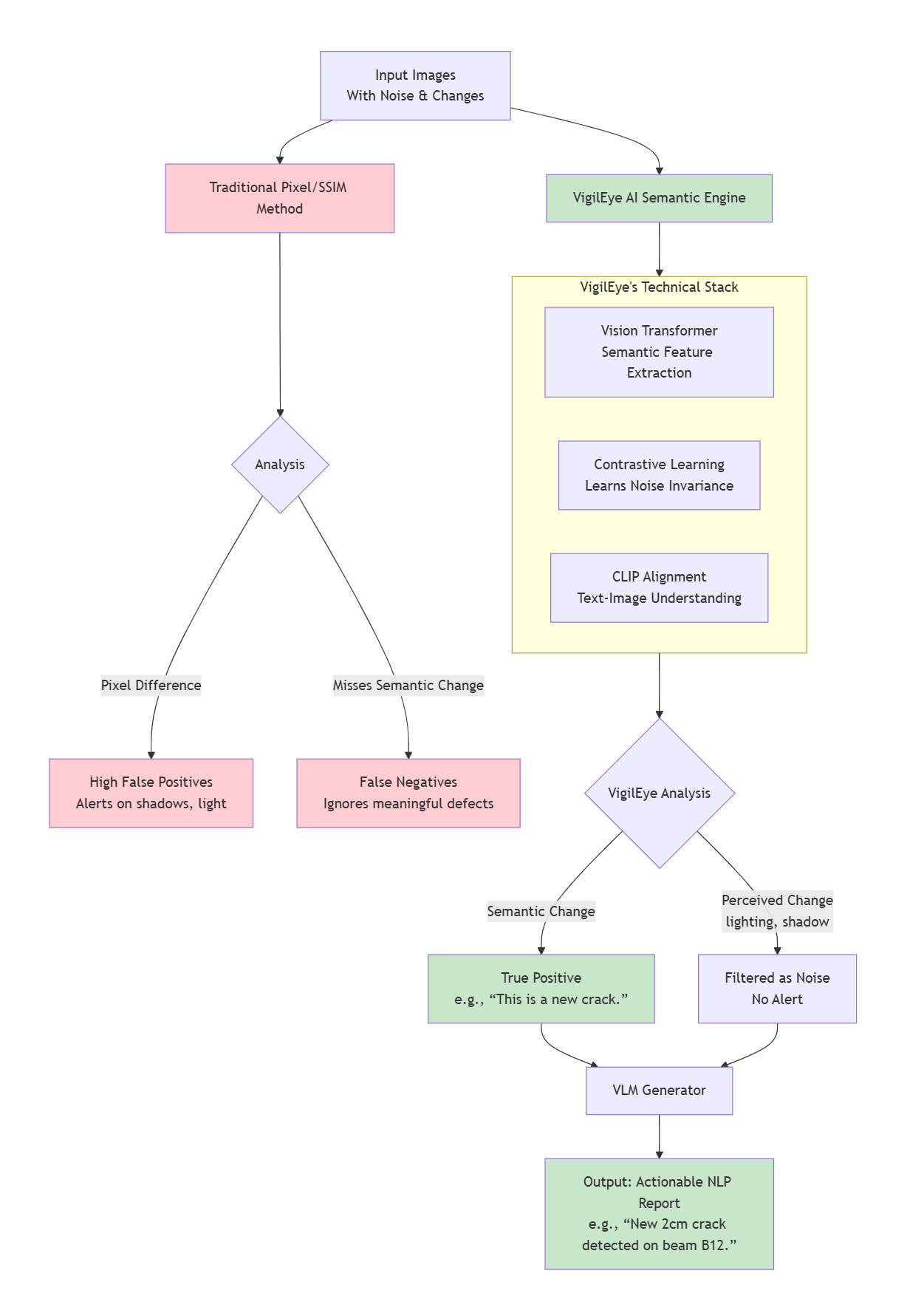
**Modular & API-First:** Containerized microservices (ViT, CLIP, VLM) scale independently via Kubernetes. FastAPI endpoints (/compare, /classify, /report) ensure seamless system integration.  
**Deployment Flexibility:** Runs efficiently on AWS, NVIDIA Jetson, or hybrid setups, optimized through ONNX/TensorRT for edge inference.  
**Cross-Domain Adaptability:** CLIP based few shot learning allows quick adaptation to new industries by fine-tuning only classifier heads.  
**Data Architecture:** Combines PostgreSQL for metadata and S3 for image storage, supporting large-scale batch inference and historical trend analysis.

**Why This Works**

**Semantic Understanding > Pixel Difference:** ViT encodes visual concepts, detecting true structural or semantic changes (e.g., cracks, dents) while ignoring lighting and noise.  
**Contrastive Robustness:** SimCLR pretraining ensures invariance to illumination, angle, and texture variations minimizing false alarms.  
**Multimodal Intelligence:** CLIP fuses vision and language for zero-shot classification and natural language change summaries.  
**Continuous Learning:** Active learning pipelines integrate new labeled data, improving accuracy and domain specific performance over time.  
**Foundation Model Leverage:** Built atop ViT + CLIP, VigilEye inherits large scale visual generalization and explainability.

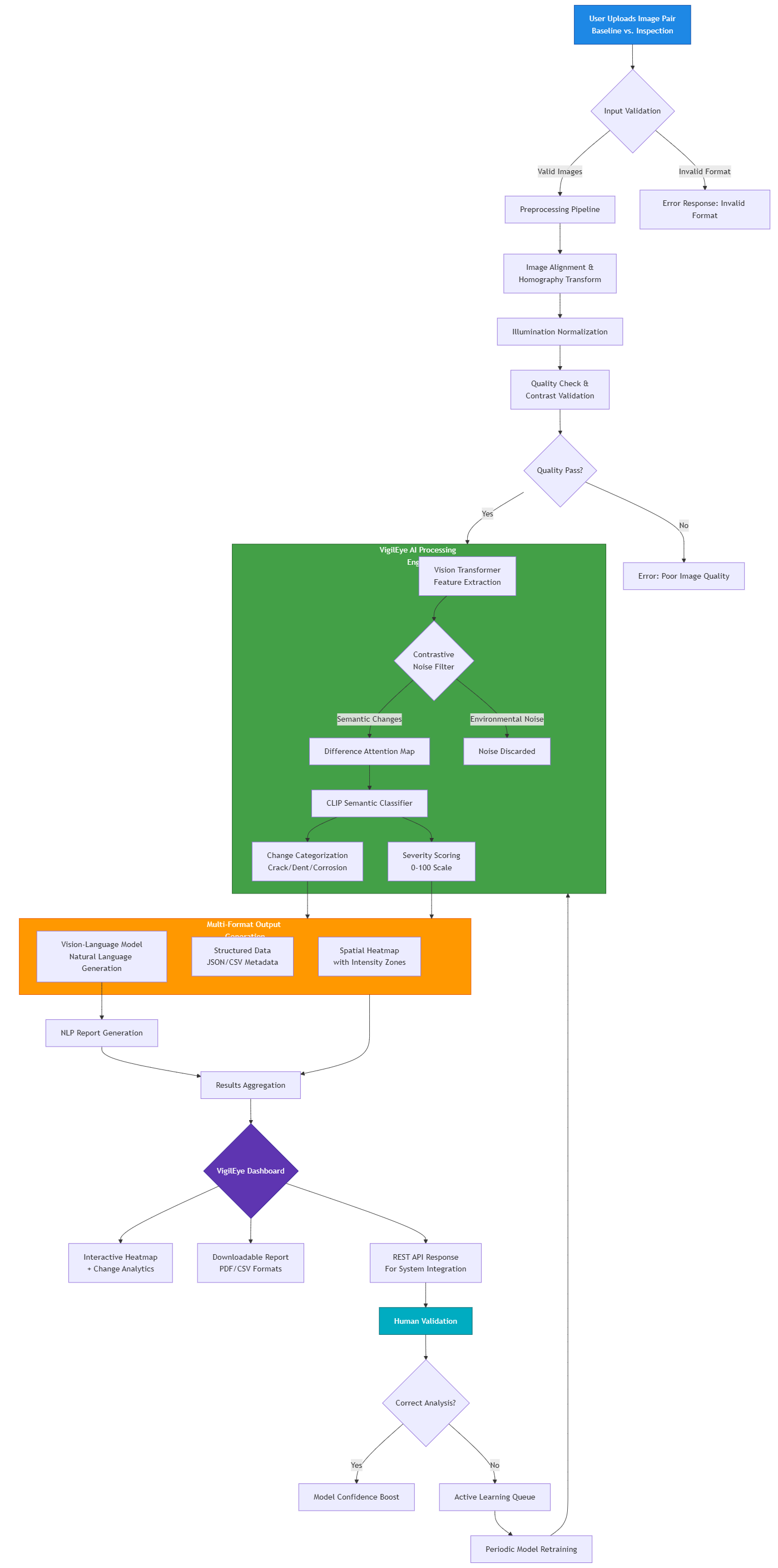


**Scalable Architecture**

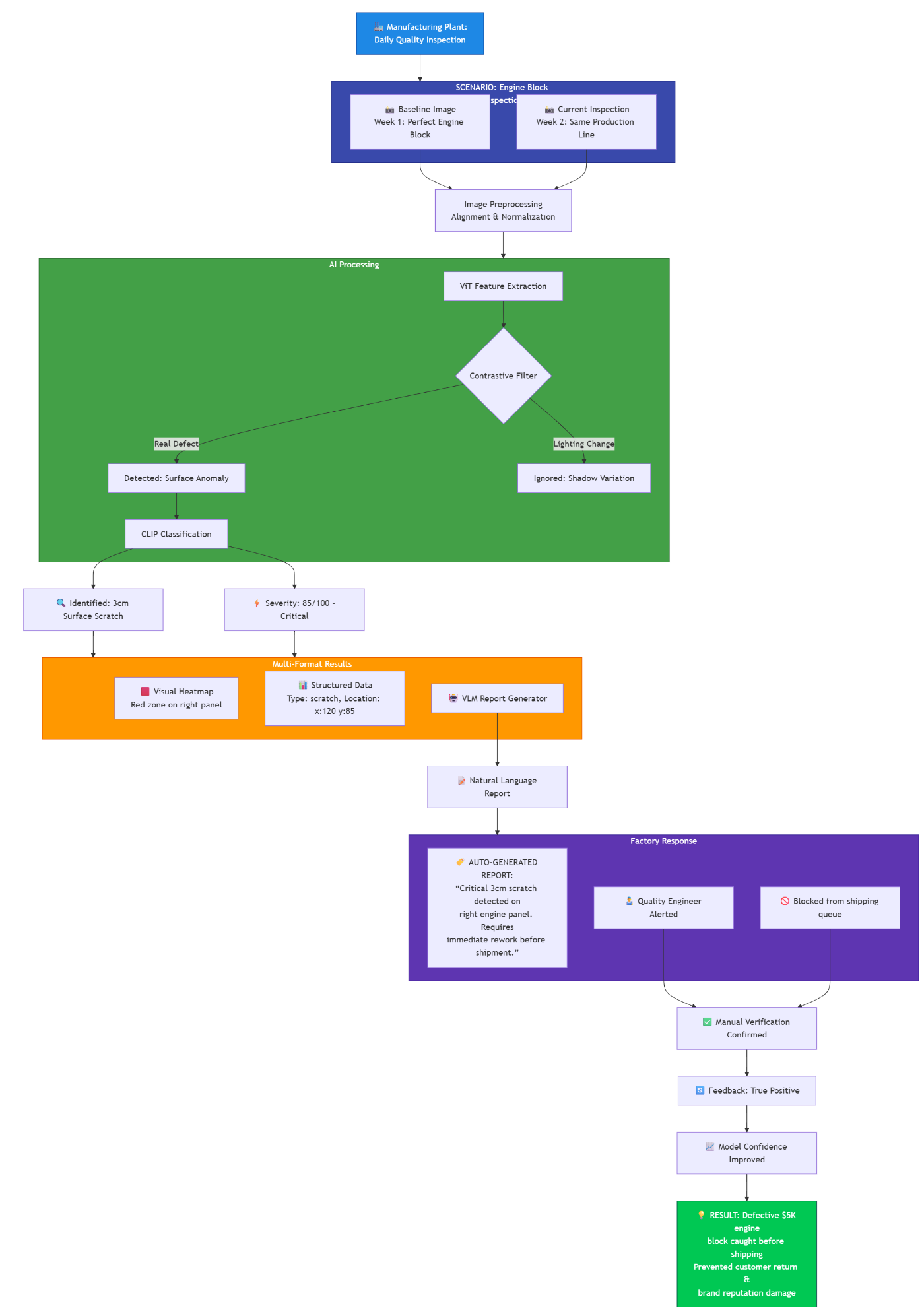


**"Why It Works" Technical Rationale Flowchart**

**4.Workflow: ENTIRE PROTOTYPE WORKFLOW(IMAGE IN THE NEXT PAGE):**



**WORKING SCENARIO:**



**Case-Based Explanation:**

Case 1: Manufacturing Quality Inspection

Scenario: A car factory wants to detect dents or missing components on car doors across production lines.

1. Input: The system receives a reference image of the car door (ideal state) and a new image from the production line camera.
2. Feature Extraction: ViT encodes both images into high dimensional semantic embeddings representing the door’s structure.
3. Noise Filtering: Contrastive learning ensures minor lighting changes or camera angle shifts do not trigger false alerts.
4. Difference Computation: Latent-space subtraction highlights regions where the structure deviates.
5. Classification: CLIP identifies the change as a “small dent near upper left panel” rather than irrelevant shadow or glare.
6. Report Generation: BLIP produces a human-readable summary:

“Minor dent detected on upper-left door panel; no other structural anomalies observed.”

1. Output: Annotated heatmap, JSON report, and textual summary are delivered to the dashboard for inspection approval.

Case 2: Brand Compliance

Scenario: A beverage company needs to verify that labels on bottles are correctly positioned.

1. Input: Reference image of correctly labeled bottle vs. image from production camera.
2. Feature Extraction: ViT embeddings capture the shape, logo position, and color patterns.
3. Noise Filtering: Variations in lighting or bottle rotation are ignored thanks to contrastive pretraining.
4. Difference Computation: ΔF identifies any displacement or deformation in the logo region.
5. Classification: CLIP categorizes the change as *“logo misalignment”*.
6. Report Generation: Natural-language summary:

“Logo displaced 5mm to the right on top section; immediate correction recommended.”

1. Output: Heatmap highlights misaligned region; JSON log stored for compliance auditing.

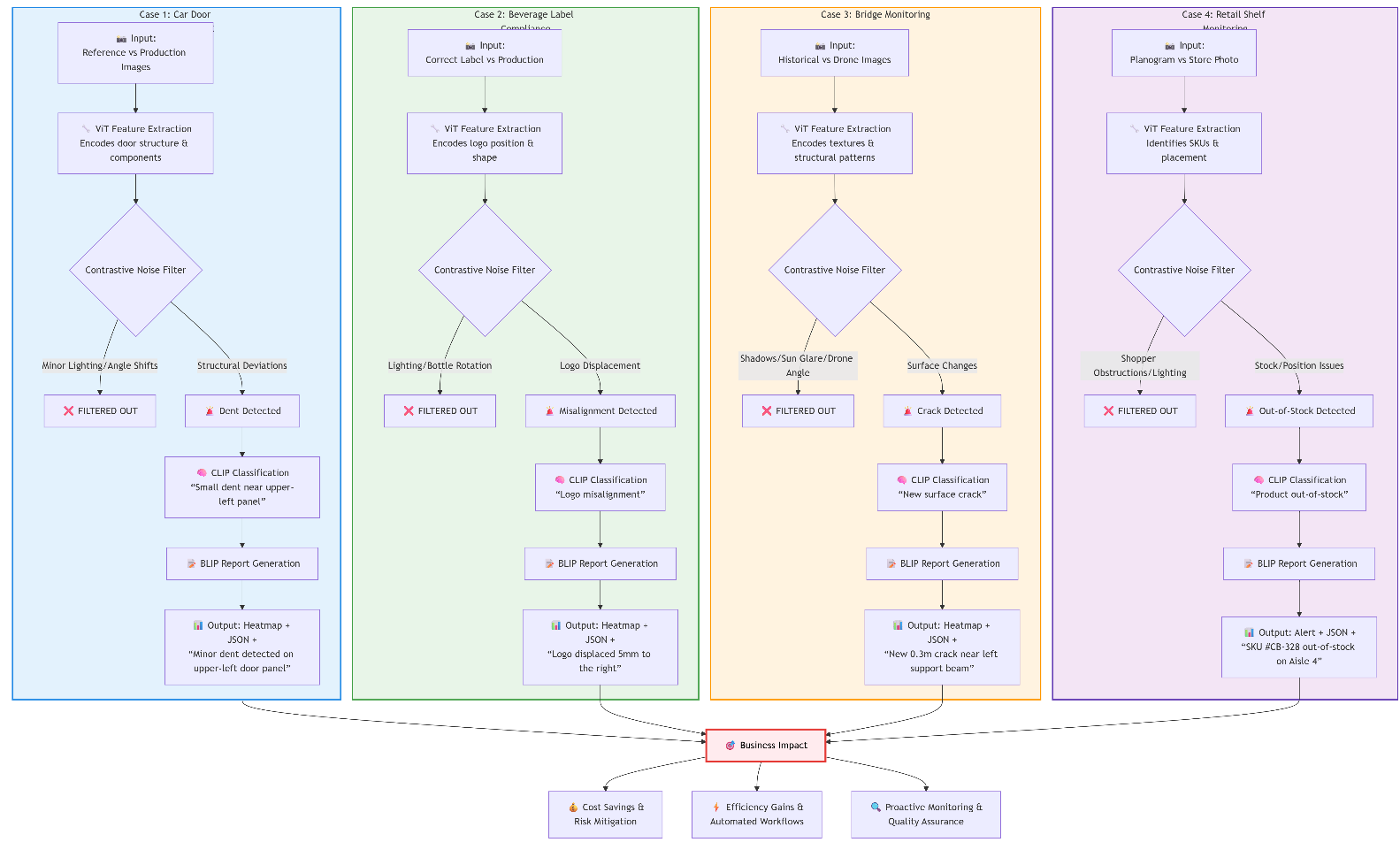
Case 3: Infrastructure Monitoring

Scenario: A city authority wants to detect cracks or damage on a bridge surface from drone images.

1. Input: Historical reference image of the bridge vs. recent drone footage.
2. Feature Extraction: ViT encodes textures, edges, and structural patterns.
3. Noise Filtering: Shadows, sun glare, or drone angle differences are ignored.
4. Difference Computation: Latent-space subtraction reveals cracks or unusual surface changes.
5. Classification: CLIP identifies change as “new surface crack near left support beam”.
6. Report Generation: Summary produced:

“New surface crack detected near left support beam, approx. 0.3m in length; requires structural review.”

1. Output: Annotated heatmap, JSON record, and textual summary sent to civil engineers for action.



**5)Technological Stack:**

**Backend**

* Python: Core language for AI pipeline, image processing, and API orchestration, leveraging its rich machine learning ecosystem.
* FastAPI: Provides low-latency, asynchronous API endpoints for image comparison, classification, and report retrieval. Supports real-time inference and integration with other systems.
* OpenCV: Handles image pre-processing, alignment, filtering, and generation of annotated heatmaps for visualization.
* PyTorch: Framework for implementing Vision Transformers, contrastive learning models, CLIP-based classification, and vision-language reporting. Supports GPU acceleration.
* Docker: Containerizes backend services to ensure reproducibility, version control, and easy deployment across cloud or edge platforms.

**AI Models**

* Vision Transformer (ViT): Encodes images into high-dimensional semantic embeddings, capturing object structure, layout, and context.
* Contrastive Learning (SimCLR-style): Pretraining module ensures invariance to illumination, camera angle, motion blur, and minor environmental noise.
* CLIP (Vision-Language): Maps visual difference embeddings to natural-language semantic categories such as cracks, dents, or misalignments. Enables zero-shot and few-shot adaptation across domains.
* BLIP / MiniGPT: Generates human-readable textual summaries describing the detected changes, providing explainable insights for human users.

**Frontend**

* React.js: Builds interactive dashboards to visualize annotated images, heatmaps, and inspection reports in real-time.
* Plotly.js / D3.js: Produces dynamic, interactive visualizations of change regions, severity maps, and temporal trends.
* Tailwind CSS: Ensures clean, responsive, and lightweight styling for the dashboard interface.

**Database & Storage**

* PostgreSQL: Stores structured metadata such as classification results, severity scores, and report logs for analysis.
* MongoDB (Optional): Handles unstructured JSON outputs or semi-structured logs from inference pipelines.
* S3 / Cloud Storage: Efficiently stores large image datasets, historical image logs, and training data for batch processing or model retraining.

**Deployment & Optimization**

* Cloud Platforms (AWS / Azure / GCP): Enables scalable inference with high-throughput processing for enterprise applications.
* Kubernetes: Orchestrates containerized microservices, allowing independent scaling of ViT, CLIP, and VLM modules.
* ONNX / TensorRT: Optimizes PyTorch models for low-latency, high-performance inference on both cloud GPUs and edge devices.
* Edge Devices (NVIDIA Jetson / Industrial AI Cameras): Supports on-site, real-time inspection without cloud dependency.
* Active Learning Pipelines: Continuously retrains models with new labeled data to improve accuracy and domain adaptation.
* Multi-View & Temporal Attention Modules: Aggregates information from sequential images, drone surveys, or CCTV time-lapses.
* CI/CD & Version Control: Git + GitHub Actions for continuous integration, testing, and deployment of system updates.

**6)Impact of VigilEye AI:**

**Operational Impact**

* Detects meaningful changes, reducing false positives and missed defects.
* Automates inspections, saving time and labor.
* Provides real-time heatmaps, severity scores, and textual reports for quick decisions.

**Cross-Domain Usability**

* Manufacturing: cracks, dents, missing components.
* Brand Compliance: logo or label misplacements.
* Infrastructure & Environment: structural degradation, surface changes.
* Minimal retraining needed for new domains.

**Explainability & Human-AI Collaboration**

* Natural-language summaries for technicians and auditors.
* Visual heatmaps highlight exact regions of change.
* Transparent and interpretable AI outputs.

**Business & Societal Impact**

* Reduces inspection costs and operational errors.
* Enables regulatory compliance with audit-ready reports.
* Enhances safety and sustainability by early detection of defects.

**7)Challenges, Risks & Assumptions:**

**Challenges**

* Labeled data needed for fine-tuning semantic change categories.
* Domain adaptation required for different industries.
* Handling large viewpoint or temporal variations.
* Moderate GPU resources required for ViT and vision-language inference.

**Risks**

* Occasional false positives or negatives due to occlusion or lighting.
* Complexity from integrating multiple AI modules.
* Edge devices may limit inference speed.
* Sensitive images require secure storage and transmission.

**Assumptions**

* Input images are roughly aligned and of acceptable quality.
* Minimal occlusion in inspection images.
* Moderate compute resources available.
* Some labeled data available for semantic fine-tuning if needed.