Property Risk AI Evaluation System

AI-Driven Architecture for Multi-Domain Risk Assessment

- **The Business Problem:** Within a legacy .NET 3.5 MVC application backed by Microsoft SQL Server, multi-record operations (e.g., bulk updates, multi-row interactions, or batch workflows) take several minutes to complete, creating slow and frustrating user experiences.
- **The Challenge:** Develop an AI-first solution approach that significantly improves the performance and usability of multi-record operations in this environment.
- **The Answer:** Integrated OpenAI agents and n8n workflows to decouple risk assessment logic and enable parallel, domain-specific analysis.

Photo by Hirsh Mohindra on Unsplash

Executive Summary

Why We Modernized the Risk Evaluation Pipeline

Why We Modernized the Risk Evaluation Pipeline

- Legacy app too slow for large surveys
- AI needed, but timeouts made results unreliable
- Parallelization + batching required
- n8n introduced as orchestration layer
- AI domain analysis now scalable and consistent

Problem Definition

The Core Challenges

- **Single-Threaded Processing:** Legacy system executed entire survey analysis in one pass, blocking users and overloading resources.
- **Database Round Trips:** User experience was poor because the amount of processing and database round trips became time consuming.
- **Computational Dependencies:** Any changes to the questionnaire would require backend computational changes This made the system brittle .
- **User-Facing Failures:** Any backend error surfaced directly to users, reducing trust and usability.

Photo by Crystal Kwok on Unsplash

Constraints of the Legacy System

Why Fixing the Code Wasn't Enough

- **Synchronous MVC Limitations:**.NET 3.5 MVC lacked async capabilities, causing the UI to freeze during heavy operations.
- **Direct SQL Writes Blocked Requests:** Data writes were part of the same cycle as user input, introducing potential delays and lockups.
- **No Background Execution Support:** Absence of workers or task queues meant long-running tasks impacted overall performance.
- **Tight Coupling of Logic and UI:** Business logic was hardcoded in the controller layer, making maintainability difficult.

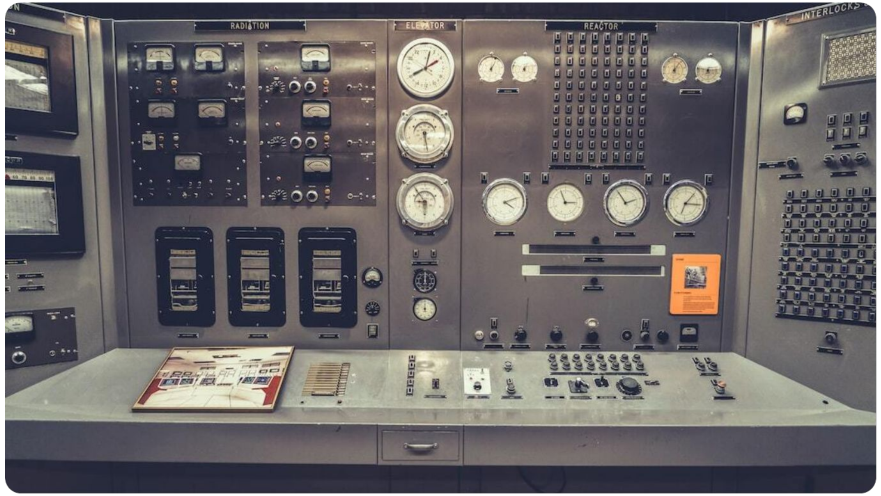


Photo by Dan Meyers on Unsplash

Solution Goals

What the New Architecture Needed to Achieve

- **Asynchronous Architecture:** Decoupled AI analysis from the MVC layer to improve responsiveness and isolate computation.
- **Parallel Domain Processing:** Each risk domain is analyzed independently, enabling efficient and scalable evaluation.
- **Reliable, Schema-Compliant Output:** Introduced validation layers to ensure all AI responses produce structured JSON with strict compliance.
- **Minimal Legacy Disruption:** Changes inside the MVC application were kept minimal, focusing modernization externally.

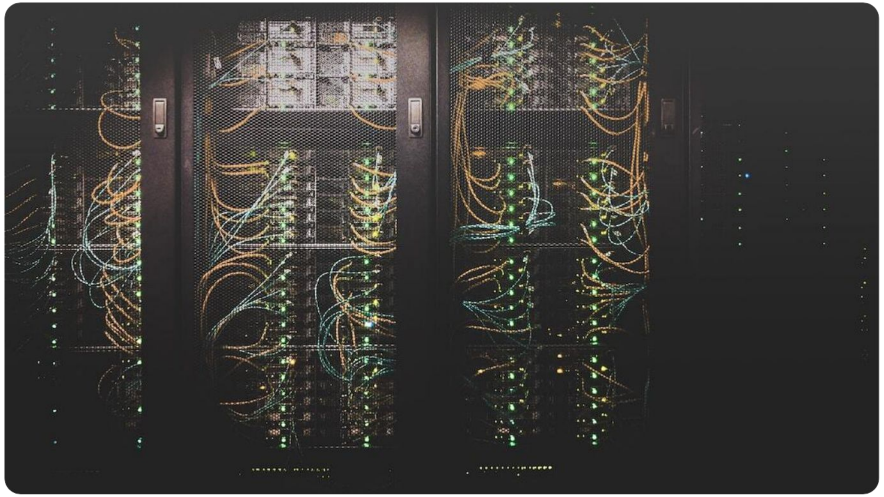


Photo by Taylor Vick on Unsplash

AI-First Philosophy

How AI Shaped the Architecture

Focused Domain Reasoning
AI performs better when prompts are scoped to a specific domain with clear inputs and expectations.

Avoiding Overload Failures
Single, large AI prompts led to hallucinations, token overflow, and inconsistent results.

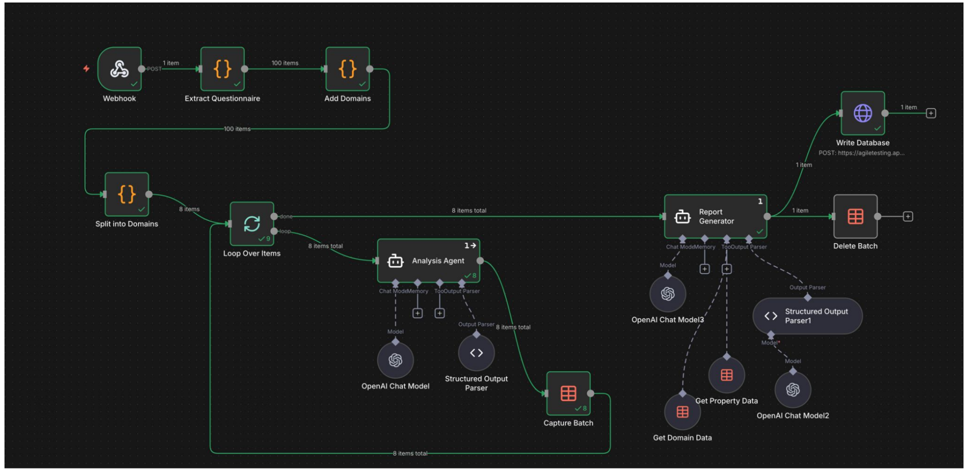
Batching + Schema Enforcement
Batched domain evaluations with strict schema outputs improved accuracy and downstream utility.

Delegated Intelligence
The orchestration layer (n8n) manages control flow, while AI handles targeted reasoning only.

High-Level Architecture

From MVC to AI-Powered Processing

- **MVC Triggers n8n Webhook:** Legacy MVC app initiates the process by posting raw survey JSON to an n8n webhook.
- **n8n Orchestrates Analysis:** JSON is cleaned, grouped by domain, and sent through a loop to OpenAI agents for evaluation.
- **Results Merged and Stored:** Processed outputs are merged into a final report and stored via a separate background workflow.
- **UI Renders Summary Only:** MVC UI now simply renders the final risk assessment, offloading all analysis to the backend.



Workflow 1 Overview

Webhook JSON → AI Analysis (Batch Mode)

- **Webhook Entry Point:** User-submitted surveys are captured via a webhook, providing an API-like intake for varied JSON formats.
- **Survey Normalization:** Unstructured input is cleaned, standardized, and reformatted into consistent question/answer pairs.
- **Domain Clustering:** Questions are grouped by risk domain, forming batches for AI evaluation.
- **Parallel AI Calls:** Each domain is independently analyzed by an OpenAI agent and results are compiled.
- **Merged Final Output:** Domain responses are aggregated into one unified JSON report for downstream use.

Workflow 2 Overview

Risk Evaluation JSON → SQL Writer

JSON Intake
Final AI-evaluated report is passed to a second workflow focused on storage.

Relational Preparation
Flattened and organized into structured formats compatible with SQL tables.

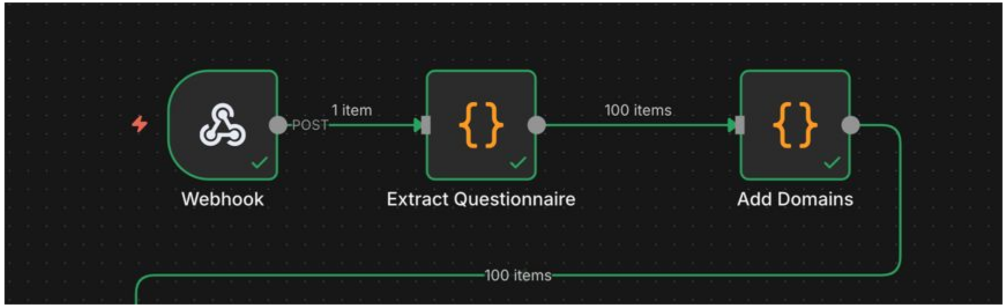
Multi-Table Inserts
Domain scores, weaknesses, and mitigation steps are stored in normalized relational tables.

Asynchronous Operation
Runs as a background task, isolating UI response times from database latency.

Node-Level Walkthrough

- **Primary Intake Point:** Webhook acts as the gateway for receiving survey JSON from the MVC application.
- **Payload Validation:** Immediately checks the JSON structure for syntactic correctness before workflow processing.
- **Async Pipeline Initiation:** Transfers the data into n8n's asynchronous processing sequence, decoupling UI delays.

Security is protected by authentication (demo uses a username and password, production would use an API secret key



Database Model Overview

Storing Final Risk Results

- **Normalized SQL Structure:** Final report is decomposed into multiple relational tables supporting full queryability.
- **Job-Level Indexing:** Each submission is stored as a Job record with foreign key links to all domain entries.
- **Separate Domain & Mitigation Tables:** Domains, weaknesses, and mitigation steps are split to enable structured analysis.
- **Analytics-Ready Design:** Model supports trend tracking, dashboards, and cross-property analytics in BI tools.

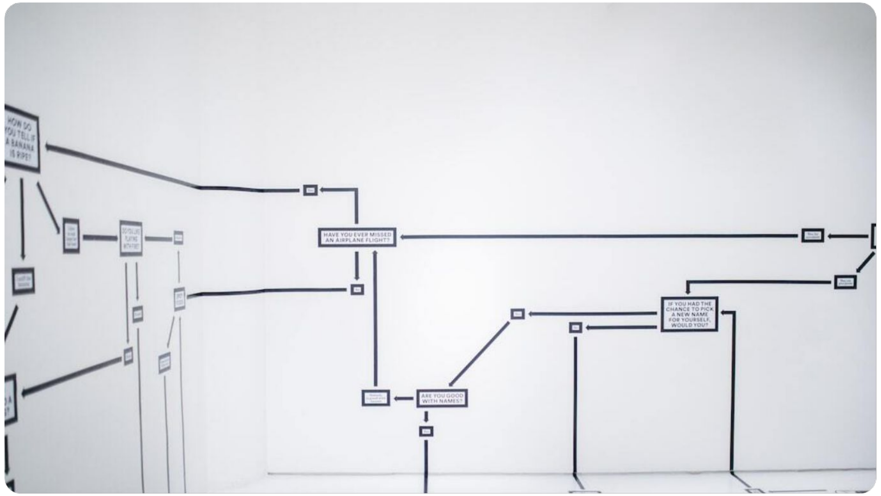

HAVE YOU EVER MISSED AN AIRPLANE FLIGHT?
ARE YOU GOOD WITH NAMES?
IF YOU HAD THE CHANCE TO PICK A NEW NAME FOR YOURSELF, WOULD YOU?
HOW DO YOU TELL IF A BANANA IS RIPE?

Photo by Hanna Morris on Unsplash

Operational Benefits

Why the New Architecture Works Better

Batch Processing
Domains are processed independently in a batch, improving speed and fault isolation.

Timeout-Free Execution
No monolithic prompts mean AI runs are shorter and more consistent.

Decoupled SQL Writes
Storage runs asynchronously, preserving UI responsiveness and eliminating bottlenecks.

Schema-Validated Outputs
All AI results are forced into a clean format, reducing parsing errors and manual corrections.

Risks & Trade-Offs

What We Accept to Gain Scalability

AI Model Dependency
Reliance on OpenAI agents introduces external service risk and requires prompt maintenance.

Workflow Complexity in n8n
Centralized orchestration demands careful monitoring and ongoing tuning.

Eventual Consistency
Asynchronous writes may delay final database visibility, trading speed for isolation.

Rule Maintenance Burden
Domain classification logic must evolve with survey changes to preserve accuracy.

Future Enhancements

Where We Can Grow Next

Voice-Based Survey Intake
AI agents that clients can speak to for dynamic, real-time data collection.

AI-Driven Insurance Alignment
Models trained to compare risks against policy guidelines for coverage optimization.

Follow-Up Automation
System-generated clarifications and deeper analysis prompts for incomplete or risky responses.

Cross-Property Analytics
Pattern discovery across risk profiles to guide portfolio-level decisions and mitigation strategies.

Conclusion

A Stable, Scalable, AI-Powered Risk Platform

- **Legacy System Preserved:** Modernization enhanced functionality without disrupting existing MVC workflows.
- **Modular AI Integration:** Domain-specific AI agents support parallel processing and consistent evaluation.
- **Reliable and Responsive:** Decoupled workflows reduce timeouts and maintain fast UI feedback.
- **Extensible and Maintainable:** Architecture supports future domain additions and evolving business requirements.

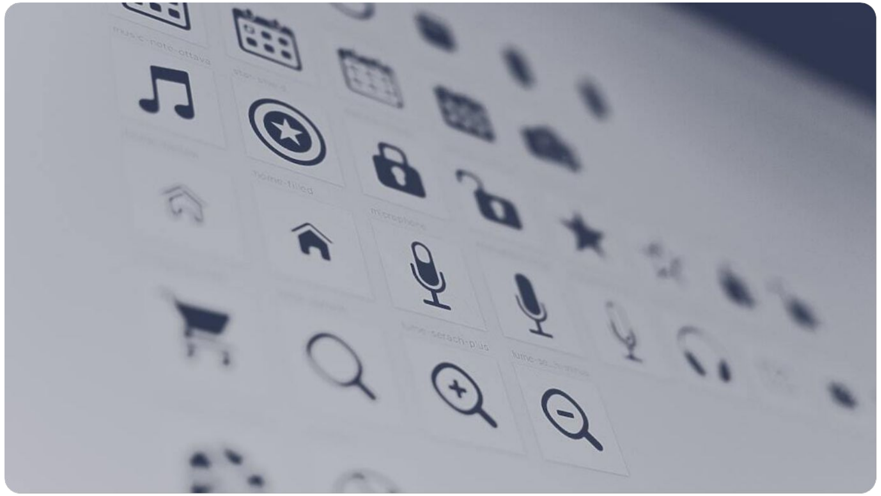


Photo by Harpal Singh on Unsplash