MICROSOFT POWER PLATFORM · PORTFOLIO CASE STUDY

Client Project Intake
& Delivery Prep Agent

Engagement Hub — a multi-agent Copilot Studio system that turns unstructured client submissions into governed Engagement Requests, AI-analysed risk, discovery briefs and Word handoff documents, with human review and audit logging built in.

★ **2nd Place · Microsoft Agent Academy Live Hackathon · Operative Track**

4	7	8+	1
COORDINATED AGENTS PARENT + SPECIALISTS	CORE DATAVERSE TABLES	POWER AUTOMATE FLOWS	SYSTEM OF RECORD DATAVERSE

Leila Marchant
Power Platform & Copilot Studio Developer

AZ-900 · AI-900 · Google Generative AI
Winchester, UK · Remote / Hybrid
leilamarchant.co.uk

ES Executive Summary

Client work rarely arrives in a clean, structured form. Briefs, contracts, statements of work and email threads land in different places, in different shapes, and a consultancy then has to triage each one consistently, surface risk early, route it to the right service offering and hand a delivery team enough context to start well. Done manually, this is slow, uneven and difficult to audit.

Engagement Hub replaces that process with a governed, multi-agent business application built on the Microsoft Power Platform. A parent **Engagement Hub Agent** orchestrates the journey; a connected **Contract Analysis Agent** analyses submitted commercial documents; and a **Delivery Prep Child Agent** produces the downstream discovery brief. Eight-plus decoupled Power Automate flows carry out the deterministic work — document analysis, duplicate-safe record creation, document generation, notifications and logging — while Dataverse acts as the single system of record. A model-driven Power Apps app gives internal teams an operational view of every submission, analysis and engagement.

The architecture is deliberately more than a single request-and-respond demo. AI findings are stored as structured operational data rather than treated as decisions; an idempotent notification flow prevents duplicate Teams alerts; duplicate Engagement Requests are detected and returned rather than recreated; and a deliberate conversational handoff keeps a human in control before delivery preparation begins.

KEY DIFFERENTIATOR

Governance is designed in, not bolted on. Pattern selection across agents, topics and flows; duplicate-safe business actions; correlation-ID tracing; a structured Automation Log event taxonomy; and a controlled human-in-the-loop handoff are core design decisions — informed by a UX background that insists a user never has to see a Dataverse GUID.

Technology Stack

- Copilot Studio
- Power Automate
- Dataverse
- Model-Driven App
- AI Prompts · Structured JSON
- Microsoft Teams · Adaptive Cards
- SharePoint
- Word Templates
- Environment Variables · ALM

01 Business Problem & Design Goals

Client intake in a consultancy is a governance problem disguised as an admin problem. The failure modes are predictable.

- Requests arrive as unstructured briefs, documents and emails, with no consistent intake shape.
- Risk and missing information are spotted late, often after work has been scoped or committed.
- Duplicate engagement records accumulate when the same brief is submitted or processed twice.
- Delivery teams inherit incomplete handoffs and have to reconstruct context from scratch.
- There is no reliable, queryable trail of what was analysed, what was decided and by whom.

The design goal was a solution that feels simple and conversational at the point of intake, while creating stronger control, traceability and operational discipline behind the scenes.

Design Principles

PRINCIPLE	WHAT IT MEANS IN PRACTICE
Dataverse as system of record	Every submission, analysis, engagement and automation event is stored as structured data — not left in chat history or flow run logs.
Right pattern for the job	Connected agents, child agents, controlled topics and agent flows are each used where they fit, rather than using agents everywhere.
Governance-first architecture	Duplicate prevention, idempotency, correlation IDs and structured error capture are core decisions, not later additions.
Human-in-the-loop by design	The system recommends and prepares; people remain accountable for routing, review and delivery decisions.
GUID-free user experience	Users work with friendly references — SUB-0002, CA-0014, ENG-0020 — and the system resolves them to row IDs behind the scenes.

02 Multi-Agent Architecture

The build is intentionally modular. Copilot Studio handles conversation and orchestration; Power Automate handles deterministic actions; Dataverse holds authoritative state. Each part has one clear responsibility.

Component Responsibilities

COMPONENT	ROLE	LAYER
Engagement Hub Agent	Parent orchestrator — greets the user, routes intent, hands off to specialist agents and controlled topics.	User-facing
Contract Analysis Agent	Connected specialist agent for contract, SOW, RFP and commercial-document analysis, with a legal-advice guardrail.	Specialist
Delivery Prep Child Agent	Generates the discovery brief and prepares the delivery handoff once an Engagement Request exists.	Specialist
Power Automate	Analysis, resolution, creation, document generation, notification and logging — delivered as focused flows.	Orchestration
Dataverse	Authoritative system of record — submissions, analysis results, engagements, review status, outputs, audit trail.	Data layer
Teams Adaptive Cards	Triage and human-review notifications, with the decision context persisted as data rather than a chat message.	Notification
Model-Driven App	Operational view — records, lifecycle status and governance charts for internal teams.	Back office

Lifecycle Sequence

1. The user submits a client document; the Contract Analysis Agent runs the analysis flow and stores a structured Analysis Result in Dataverse.
2. Analysis context — risk count, confidence, human-review recommendation, correlation ID — is carried forward through global variables.
3. The user selects a friendly service route; a resolve flow maps it to the underlying Service Offering record.
4. A controlled confirmation topic shows an adaptive card and asks the user to confirm before any record is created.
5. The create flow checks for an existing matching Engagement Request and returns it if found, or creates a new one.
6. The user chooses to prepare the delivery handoff; the Delivery Prep Child Agent generates the discovery brief and updates Dataverse.
7. Notification flows post Teams alerts where human review is required, and every step is written to the Automation Log.

03 Pattern Selection: Agents, Topics & Flows

The most important architectural work was not adding agents — it was deciding which pattern fits which job. Using generative orchestration everywhere adds risk; using it nowhere wastes the platform.

PATTERN	USED WHEN…	IN THIS BUILD
Connected agent	The task has a strong, self-contained domain boundary and its own guardrails.	Contract Analysis Agent
Child agent	The task is a focused sub-task within the same business process.	Delivery Prep Child Agent
Controlled topic	Behaviour must be deterministic — no generative improvisation at a decision point.	Engagement Request Confirmation
Agent flow	Records must be created, retrieved or updated reliably.	Create / Resolve / Analyse flows
Backend cloud flow	Automation should be event-driven and run without a user present.	Triage & human-review notifications

DESIGN DECISION

Record creation is a *controlled topic*, not an agent action. The system must never create a business record until a person explicitly confirms — that is not a decision generative orchestration should make on its own.

04 Dataverse: State as an Audit Trail

The data model was designed first. Submissions, analysis, engagements and every automation event have a home in Dataverse, so the system is observable without relying on flow run history — which is ephemeral and unsuitable for governance.

Core Tables

TABLE	PURPOSE	CATEGORY
Submissions	Inbound client briefs and documents — the entry point for the intake journey.	INTAKE
Analysis Results	Structured output of document analysis: risk flags, gaps, confidence, review status, correlation ID.	AI
Engagement Requests	Authoritative engagement record, linked to its submission, analysis and service offering.	LIFECYCLE
Service Offerings	The catalogue of service routes a request can be matched to.	REFERENCE
Qualification Criteria	Criteria used to assess and route submissions to the right offering.	REFERENCE
Client Contacts	The people and organisations behind each submission.	IDENTITY
EH Automation Logs	Structured event log for every flow action — the operational audit trail.	TRACEABILITY

DESIGN DECISION

A dedicated **Analysis Results** table replaced an earlier legacy table partway through the build, with a strict rule that all new flows use the new schema. Resisting the temptation to keep wiring the old table in was what kept the data model clean enough to reason about under deadline.

05 Contract Analysis: AI With Structured Output

The Contract Analysis Agent turns a free-form commercial document into structured, queryable operational data — and never into a decision.

The analysis flow retrieves the submission, extracts the document text, runs the document-analysis prompt, parses the structured JSON result and writes it back to Dataverse. Each run captures:

- Risk flags identified in the document
- Missing or ambiguous information
- A confidence score for the analysis
- A human-review recommendation
- A plain-language analysis summary
- A correlation ID for end-to-end tracing

The agent operates behind an explicit **contract-approval and legal-advice guardrail**: it can surface risks and gaps, but it will not approve terms, give legal advice, or make commercial or go/no-go decisions.

ENTERPRISE PATTERN

AI output is treated as *evidence to be reviewed*, not an outcome to be trusted. Storing the confidence score and a human-review flag alongside the findings means the platform can route uncertain analyses to a person automatically — the model's uncertainty becomes a governance signal.

06 Power Automate: Decoupled Agent Flows

Rather than one large orchestration, the solution uses focused flows — each with a single responsibility, easier to test, reason about and support.

FLOW	RESPONSIBILITY
EH AF – Analyse Contract Document	Extracts text, runs the analysis prompt, parses structured output, writes the Analysis Result and start/complete/fail logs.
EH AF – Resolve Service Offering	Maps a friendly service-route name to the underlying Service Offering row ID, so the user never types a GUID.
EH AF – Create Engagement Request	Checks for an existing active engagement; returns the duplicate if found, otherwise creates and links a new record.
EH AF – Generate Discovery Brief	Produces the structured discovery brief, updates the Engagement Request and writes the Word handoff document.
EH – Notify Delivery Triage Channel	Posts a Teams adaptive card to the triage channel, gated by an idempotency flag.
EH – Notify Human Review Channel	Autonomous, idempotent notification when an Analysis Result requires human review.

Supported by helper flows and child flows — including status lookups and document generation — for eight-plus flows in total across the solution.

07 Duplicate-Safe Requests & a GUID-Free UX

Duplicate-safe Engagement Requests

Before creating an engagement, the create flow checks for an existing active Engagement Request that matches on submission, analysis result and service offering. If one exists, the flow returns it instead of creating a second record — and logs the event as a successful, governed outcome rather than an error.

Users never see a GUID

A UX background drove a rule applied across the whole solution: people work with business-friendly references — Submission Number, Analysis Number, Engagement Number — while the flows resolve those to Dataverse row IDs behind the scenes using a dedicated resolve flow and a set of global context variables.

DESIGN DECISION

Resolving friendly references costs an extra flow and some duplicate-match handling. It is worth every minute: a thirty-six-character row ID on screen is technically correct and completely inhuman. The difference between a working agent and a credible one is largely this kind of UX discipline.

08 Delivery Prep: Conversational Handoff

The most important architectural decision in the build was something deliberately *not* built: a fully autonomous, Dataverse-triggered generation flow.

In the final design, once an Engagement Request exists the agent asks the user what to do next. Only when the user chooses **Prepare delivery handoff** does the Delivery Prep Child Agent generate the discovery brief, save it back to Dataverse, produce a Word handoff document in SharePoint, and surface the result in a polished adaptive card.

The card itself is part of the architecture: it shows the child agent ran, the brief was generated, Dataverse was updated, human-review status, and the correlation ID — making the multi-agent design visible rather than merely described.

WHY THIS MATTERS

A visible conversational handoff keeps a human in control before downstream artefacts are generated, proves the orchestration in a short demo where a hidden trigger would prove nothing, and shows the right production order: the controlled version first, with event-driven automation as a later evolution once review gates exist.

09 Autonomous Review Alerts & Idempotency

When an Analysis Result requires human review, a standalone, autonomous flow posts a Teams adaptive card to the review channel. It notifies only — it never approves, rejects or assigns.

The card surfaces the analysis number, risk flags, AI confidence, recommended action, correlation ID and a governance note stating explicitly that the notification does not approve contracts, pricing or commercial terms.

Idempotency that actually fires

Editing the same record twice initially produced a second Teams message. The fix tightened the duplicate check to query the Automation Log for a prior successful `human_review_notification_sent` event against the same analysis row before posting. Once corrected, a repeated edit produced no duplicate alert — and the duplicate-prevention decision was itself visible in the logs.

LESSON

An idempotency flag is only as good as the check that reads it. The control worked only once the event name was written as an exact, typed value and the lookup matched on event, success and related row together — not on any single field.

10 Governance, Audit & the Automation Log

The least glamorous table is the one worth defending hardest. Every flow writes to **EH Automation Logs**, and a structured event taxonomy makes the log something you can build views and governance charts on — not just noise that says "something ran".

Event Taxonomy (selected)

EVENT NAME	MEANING
`contract_analysis_started`	Analysis flow began for a submission.
`contract_analysis_completed`	Analysis succeeded and an Analysis Result was written.
`contract_analysis_failed`	Analysis failed; logged with severity Error.
`engagement_request_created`	A new Engagement Request was created.
`engagement_request_duplicate_detected`	An existing engagement was returned instead of duplicating.
`human_review_notification_sent`	A review alert was posted to Teams.

Each row also records success, severity, related submission / analysis / engagement row IDs, flow and agent names, a user-facing message and a correlation ID — so any event can be traced end-to-end without flow-admin access.

KEY DIFFERENTIATOR

None of this appears in a demo's "wow" moment. All of it appears the first time something goes wrong in production and someone asks: what happened, when, and did we already act on it?

11 Error Handling: Business vs Technical Failure

A failed lookup is not a technical exception. A reference that does not resolve — a typo, a deactivated record — is a business outcome the agent should relay calmly, not a reason to terminate the run.

Agent flows therefore avoid hard termination. The pattern instead:

- Resolve the reference; if exactly one record is found, set a continue flag to true.
- If zero or multiple are found, set failure-message variables and leave the flag false — and do nothing else in that branch.
- Gate the main work behind a single condition on that flag.
- Guarantee the final *Respond to agent* action runs on every path — success, business failure, or a technical error caught in Try/Catch.

Every run ends with a controlled response, the agent always has something sensible to say, and the Automation Log always has a record of what happened.

LESSON

Initialise variables at the top level only; inside conditions and scopes, use Set. And remember that a Dataverse Yes/No field that reads as "Yes" in the app may evaluate as `true` in a flow — conditions must test the value the flow actually receives.

12 Challenges Solved

1 · A publish-blocking tool conflict

Testing the delivery handoff raised a `ToolIdentifierConflict` — the same flow was exposed in two places, so Copilot Studio could not tell which to call. The fix removed the duplicate exposure, keeping the tool wired into the child agent and allowing a parent direct-call only as a deliberate fallback.

2 · Stale, cached action references

Removing the duplicate then surfaced stale reference errors. The lesson learned the hard way: Copilot Studio can cache old action references, and a tool already wired into a child agent should be repaired by a clean remove-and-re-add of the component — not by deleting and recreating a tool other components still depend on.

3 · A demo that ended on its weakest note

The happy path originally finished with a plain sentence after a journey of polished cards. The final step was rebuilt as a Discovery Brief Result adaptive card, so the proof point of the whole multi-agent story looks like a business-system result rather than a chatbot confirmation.

4 · Going beyond the happy path

Idempotent notifications, duplicate-safe creation, error capture back to Dataverse and controlled failure paths all address what happens when something goes wrong or the system runs over time — which is what production work actually looks like.

13 Testing Approach

The solution was tested against explicit scenarios, validating both the conversational behaviour and the resulting Dataverse state after each action.

- Successful document analysis and structured Dataverse write
- Analysis requiring human review — correct flag and notification
- Service-route selection resolving to the correct offering
- Confirmation gate — no record created before the user confirms
- New Engagement Request creation — happy path
- Duplicate request — existing record returned, not recreated
- Delivery handoff — discovery brief generated and saved
- Word handoff document produced and stored in SharePoint
- Human-review alert posted to Teams
- Idempotent behaviour — no duplicate alert on re-edit
- Business failure — unresolved reference handled without termination
- Automation Log written on every success and failure path

14 What This Demonstrates

This case study is structured to show the skills a Microsoft Partner expects from a developer working on enterprise Power Platform and agent engagements.

SKILL AREA	HOW IT IS DEMONSTRATED
Multi-agent architecture	A parent orchestrator coordinating a connected specialist agent and a child agent, each with a defined role.
Pattern selection	Deliberate use of connected agents, child agents, controlled topics and flows — generative reasoning only where it adds value.
Dataverse as system of record	Authoritative state, relationships and audit history — not a data store behind a chat.
Power Automate at scale	Decoupled flows, Try/Catch handling, idempotency controls and child-flow reuse.
Resilient AI integration	Structured JSON output, confidence scoring and a human-review recommendation as a governance signal.
Human-in-the-loop control	A deliberate conversational handoff so a consultant chooses when downstream work begins.
Governance & auditability	Idempotency, correlation IDs and a structured Automation Log event taxonomy.
Operational UX	Friendly references over GUIDs, adaptive cards and model-driven views that make outcomes clear.

15 What I Would Build Next

Scope was drawn deliberately. The build is production-shaped, so the next iterations are additive rather than a rebuild.

- **Automated intake channels** — a secure request form, shared mailbox and Teams entry point that create Submissions automatically and resolve client contacts.
- **Review workflow & SLAs** — reviewer assignment, decision capture, approval actions and time-based escalation so high-risk findings have a clear owner and cannot progress unnoticed.
- **Event-driven delivery prep** — once review and approval gates exist, evolve the conversational handoff into an autonomous Dataverse-triggered generation flow.
- **Production governance** — role-based access, environment-aware deployment pipelines, Managed Environments with DLP and CoE integration.
- **Insight layer** — a Power BI reporting layer over Dataverse for submission volume, risk profile, duplicate-prevention rate and delivery-prep throughput.
- **Telemetry & feedback** — Application Insights instrumentation and a lightweight satisfaction signal on generated briefs.

COMPANION MATERIAL

The full case study, architecture diagram and demo walkthrough are available at **leilamarchant.co.uk**, alongside the Access Request & Approval Agent case study and its ALM Release Checklist.