MICROSOFT POWER PLATFORM · PORTFOLIO CASE STUDY

Access Request & **Approval Agent**

Enterprise access management - conversational intake, structured approvals, AI summarisation, and autonomous SLA operations.

Leila Marchant
Power Platform & Copilot Studio Developer

AZ-900 · AI-900 · PL-900
Winchester, UK · Remote / Hybrid

6	23+	4	2
Power Automate · Flows	Dataverse · Fields	Security · Roles	Auth · Modes

ES EXECUTIVE SUMMARY

Most internal access request processes fail in predictable ways: inconsistent intake, approvals lost in inboxes, duplicate notifications, and no reliable audit trail when something goes wrong.

This project replaces that process with a structured, auditable, identity-aware business application built on the Microsoft Power Platform. Copilot Studio handles conversational intake with policy-aware gating. Six decoupled Power Automate flows manage creation, approval, resolution, and SLA monitoring. Dataverse is the single system of record - every lifecycle event, decision, timestamp, and error is written back to one business record. A model-driven Power Apps app gives internal teams a live operational view.

The architecture goes beyond a simple request-and-approve demo: idempotent watcher patterns prevent duplicate Teams notifications, a BYOM Azure OpenAI summarisation tool runs as a reusable child flow with fail-safe fallback, and a scheduled SLA reminder watcher maintains workflow health autonomously.

> **KEY DIFFERENTIATOR**
> Governance is built in - not bolted on. RBAC, state-based edit controls, Dataverse audit history, and structured error capture are core design decisions, not afterthoughts.

Technology Stack

Copilot Studio	Power Automate	Dataverse	Power Apps	Teams	Entra ID	Azure OpenAI	Key Vault	VS Code + PAC CLI

01 BUSINESS PROBLEM & DESIGN GOALS

Many internal access request processes are still inconsistent, manual, and difficult to govern. The problems are predictable:

- Users submit vague or incomplete requests with no structured intake
- Approvals happen over email or chat - no system of record
- Duplicate approval notifications sent when workflows retrigger
- Request status is invisible to the requester after submission
- No reliable way to prove who requested access, who approved it, and when
- Overdue approvals sit unnoticed - no operational follow-up

The design goal was to build a solution that feels simple and conversational for the requester, while creating stronger control, auditability, and operational discipline behind the scenes.

Design Principles

Principle	What It Means in Practice
Dataverse as system of record	Every lifecycle event stored as structured data - not in email threads or flow run history.
Decoupled orchestration	Separate watcher flows for approvals, resolution, and SLA reminders. Each with one clear responsibility.
Governance-first architecture	RBAC, state-based edit controls, idempotency, and error capture are core decisions - not additions.
Resilient AI integration	Summarisation is a child flow with fail-safe fallback. The main workflow never breaks if the model fails.
Enterprise authentication	Secure variant uses Entra ID signed-in identity - requests tied to a verified user, not typed data.

02 ARCHITECTURE OVERVIEW

The build is intentionally modular. Copilot Studio handles intake and policy gating. Power Automate handles orchestration. Dataverse holds the authoritative lifecycle state. Watcher flows operate asynchronously so the system continues to progress after the user leaves the chat.

Component Responsibilities

Component	Role	Layer
Copilot Studio	Conversational intake, input validation, policy gate, status check, Entra ID auth (secure variant)	User-facing
Power Automate	Create, approval, resolution, SLA reminder, AI summarisation - 6 focused flows	Orchestration
Dataverse	Authoritative system of record - lifecycle state, decisions, timestamps, errors, full audit trail	Data layer
Teams Adaptive Cards	Structured approval notifications - decision persisted as data, not as an email thread	Approval
Model-Driven App	Internal operational view - views, forms, lifecycle visibility for approvers and admins	Back office
Azure OpenAI / Key Vault	BYOM case summarisation with environment-variable config and Key Vault secret management	AI layer

Lifecycle Sequence

1. Copilot Studio collects the request, applies validation, and enforces the policy gate for admin-level access.
2. The Create flow generates a correlation ID, calls the AI summarisation child flow, and writes the Dataverse record.
3. The Approval Watcher detects the pending state and posts one Teams card (idempotency flag prevents duplicates).
4. The approver decision is written back to Dataverse as structured data - not stored in Teams.
5. The Resolution Watcher processes the outcome, stamps completion fields, and notifies the requester.
6. The SLA Reminder Watcher runs on a schedule - finds overdue approvals, sends reminders, escalates after a threshold.
7. The model-driven app provides live operational visibility at every stage.

03 DATAVERSE SCHEMA: STATE AS AN AUDIT TRAIL

The data model is designed first. Every workflow event - including failures - is represented by a dedicated column. This makes the entire system observable without relying on flow run history, which is ephemeral and unsuitable for governance or support purposes.

> **DESIGN DECISION**
> Writing error details back to Dataverse fields - not letting them disappear into run history - is what makes this solution supportable in production. A support engineer can investigate any case without needing flow admin access.

Key Fields by Category

Field	Purpose	Category
RequestorEmail / UPN	User-submitted email (public) or Entra UPN (secure variant)	Identity
RequestorAadObjectId	Entra object ID for trusted identity binding in the secure variant	Identity
ApproverEmail	Designated approver for this request	Identity
Status / StatusReason	Authoritative lifecycle state - drives all downstream watcher logic	Lifecycle
SubmittedOn / DecisionOn	Audit timestamps for submission and approval decision	Lifecycle
ApprovalRequestSent / SentOn	Idempotency flag - prevents duplicate Teams approval cards	Idempotency
ApproverDecision / Comments	Structured decision outcome captured as queryable data	Approval
CaseSummary	AI-generated operational summary from the BYOM child flow	AI
CorrelationId	End-to-end trace ID tying all events to a single request	Traceability
FulfilledOn / FulfillmentError	Fulfilment outcome and structured error capture	Fulfilment
ApprovalError	Approval watcher error detail - survives flow retries	Error Handling
ReminderCount / LastReminderOn	SLA reminder audit trail	SLA
EscalationCount / EscalatedOn	Escalation tracking once reminder limit is reached	SLA

04 COPILOT STUDIO INTAKE: VALIDATION & POLICY GATE

The agent topic collects a structured and consistent set of inputs before any automation begins. This prevents incomplete or ambiguous requests from entering the workflow - a significant source of operational noise in manual processes.

Structured Input Collection

- Application name and requested access level
- Business justification
- Duration in days
- Approver email

Policy Gate for High-Risk Requests

Admin-level access requests trigger an explicit policy acknowledgement step - the requestor confirms they understand they are requesting elevated access before the workflow proceeds. This adds a visible governance checkpoint without requiring a separate process.

> **ENTERPRISE PATTERN**
> In the secure authenticated variant, requestor identity is captured from the Entra ID signed-in context - not from user-typed input. This eliminates a class of data quality issues and makes the request legally attributable to a verified identity.

GetStatus Topic

A separate status-check topic allows requestors to query the current state of their submission at any point. This reduces support burden and gives users direct access to authoritative lifecycle data from Dataverse.

05 POWER AUTOMATE: SIX DECOUPLED FLOWS

Rather than one large orchestration flow, the solution uses six focused flows - each with a single, clear responsibility. This makes the architecture easier to reason about, test, and support.

Flow	Responsibility
Create (Copilot-triggered)	Validates inputs, generates a correlation ID, calls the SummariseCase child flow, creates the Dataverse record, and returns structured outputs to the bot.
SummariseCase (Child flow)	Reusable AI summarisation tool. Endpoint, deployment name, API version, and prompt are driven by environment variables. Fail-safe fallback ensures the parent flow always completes.
GetStatus	Helper flow returning current lifecycle state to Copilot Studio, enabling the status-check topic.
Approval Watcher	Monitors Dataverse for pending requests with no approval card sent. Posts one Teams card per request. Writes the decision back to Dataverse as structured data.
Resolution Watcher	Processes approved and rejected outcomes. Stamps completion fields, notifies the requestor. Try/Catch - failures written to Dataverse, not lost in run history.
SLA Reminder / Escalation Watcher	Scheduled autonomous operations layer. Scans for overdue approvals, sends reminders, increments counters, and escalates once a configurable limit is reached.

06 BYOM SUMMARISATION: REUSABLE AI WITH FAIL-SAFE

AI summarisation is implemented as a reusable child flow rather than embedded inline in the main transaction. This separation makes the component independently testable, reusable across flows, and easier to maintain.

Architecture Decisions

- Model settings are configuration, not code - endpoint, deployment name, API version, and system prompt are all environment variables
- Azure OpenAI API key stored in Azure Key Vault - not hardcoded in the flow
- If the AI call fails or throttles, the parent flow receives a safe fallback summary and continues - request creation is never blocked by an AI failure
- The child flow is callable from any other flow in the solution, not just the create flow

BYOM PATTERN

Externalising model configuration via environment variables means the AI model can be changed or upgraded without editing the flow - only the environment variable values need updating. This is the correct enterprise pattern for managing AI dependencies across deployment environments.

07 AUTONOMOUS OPERATIONS: SLA REMINDER WATCHER

The SLA watcher is what separates an agentic system from a chatbot with memory. It runs on a deliberate schedule, operates without being prompted, and maintains workflow health over time.

What It Does

- Scans Dataverse for pending requests where the approval card has been sent but no decision has been returned
- Identifies overdue requests based on submission timestamp and a configurable SLA threshold
- Sends a reminder to the approver and increments the ReminderCount field
- Updates the LastReminderOn timestamp so the next run does not send a duplicate
- Once ReminderCount reaches a configurable maximum, escalates the request and records the escalation

Every action taken by the watcher is written back to Dataverse - creating an operational audit trail of exactly what autonomous actions the system took, and when.

> **WHY THIS MATTERS**
>
> Most portfolio demos show only the happy path. The SLA watcher demonstrates what happens when a request stalls - and proves the system can maintain itself operationally without human intervention. This is the behaviour enterprise clients expect from production systems.

08 SECURITY & GOVERNANCE

Governance is implemented in layers so each control is independently visible and demonstrable in screenshots and platform recordings.

Role-Based Access Control

Role	Privileges
Requester	Create new requests and read their own records only
Approver	Read pending requests and update approval decisions
Admin / Operations	Full operational visibility and record management
Automation (Service)	Least-privilege service-level access for flow operations

State-Based Edit Controls

Request fields are locked once a record leaves draft status. This prevents users from amending what they originally submitted after the process has started - a governance requirement in any auditable access management system.

Idempotency Controls

The approval watcher uses an ApprovalRequestSent flag and a sent timestamp so it posts exactly one Teams card per request. If the flow retriggers due to a Dataverse platform event, it checks the flag first and takes no action.

Error Capture and Auditability

Watcher flows use Try/Catch/Finally patterns. Failures are written back to dedicated Dataverse fields rather than disappearing into flow run history - making the system supportable without flow admin access.

09 ALM & DEPLOYMENT

The solution is built to ship - not just to demo. Every component lives inside a named Power Platform solution and every environment-specific value is externalised. This means the exact same managed artefact tested in UAT is what gets promoted to Production.

Deployment Practices

Practice	Implementation
Solution naming	lai_AccessRequestAgent. Publisher prefix: lai_ applied consistently across all tables, columns, flows, and relationships - no Default publisher sprawl.
Managed vs Unmanaged	Unmanaged solutions for development only. Only Managed solutions are exported and deployed. Managed solutions are immutable post-import - no in-place flow edits in UAT or Production.
Environment Variables	All endpoint URLs, API keys, SLA thresholds, routing addresses, and model parameters are externalised as Environment Variables. Nothing environment-specific is embedded in a flow action.
Connection References	Every connector is abstracted behind a Connection Reference. Service identities are bound by the importing admin at import time - the flow is never hard-wired to a maker account.
Versioning	MAJOR.MINOR.PATCH.BUILD format. Versions are immutable - the same managed zip that passed UAT is promoted unchanged. PATCH and MINOR releases use Update; MAJOR releases use Upgrade.
Rollback	The previous managed solution zip is retained as a rollback artefact after every promotion. Production is never left without a validated prior state to return to.

COMPANION DOCUMENT

The ALM Release Checklist is a companion reference to this case study. It contains the full deployment pipeline (stages A–E), Definition of Done with 9 evidenced criteria, governance checklist, and the versioning decision framework - all applied to this solution.

10 AUTHENTICATION: PUBLIC DEMO vs SECURE ENTERPRISE

Authentication decisions are made explicitly. The build supports two modes depending on deployment context.

Mode	Behaviour
Public / Demo mode	No sign-in required. Requestor email collected via conversation. Low friction for recruiter demos and portfolio reviews.
Secure / Enterprise mode	Manual Entra ID authentication. Requestor identity captured from signed-in context. AAD object ID and UPN stored in Dataverse. Suitable for production enterprise deployment.

Building both modes was a deliberate portfolio decision. The public version allows anyone to test the experience. The secure version proves the architecture can be adapted for enterprise identity requirements without rebuilding the conversation flow.

11 CHALLENGES SOLVED

1. Preventing duplicate approval notifications

Solved with an explicit idempotency flag (ApprovalRequestSent) and a sent timestamp so the watcher always checks before acting - even if the Dataverse trigger fires multiple times.

2. Lifecycle architecture rather than a single flow

Separating approvals, reminders, fulfilment, and AI into focused watcher flows makes each concern independently verifiable and much closer to how enterprise systems are actually built.

3. Making AI useful without making the workflow fragile

Separating summarisation into a child flow with explicit fallback handling means the parent transaction is always reliable. The AI layer is an enhancement, not a dependency.

4. Enterprise authentication without breaking the demo

Building two explicit modes - rather than compromising one for the other - solved both requirements cleanly. Recruiters can test freely; enterprise clients see real identity-driven automation.

5. Going beyond the happy path

The SLA watcher, error capture back to Dataverse, idempotency patterns, and the secure authentication variant all address what happens when things go wrong or when the system needs to operate over time. That is what enterprise work actually looks like.

12 TESTING APPROACH

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

- Successful request creation - public variant
- Admin-level access request with policy acknowledgement gate
- Teams approval - happy path (approved outcome, fulfilment processing)
- Teams approval - rejection path (correct final state and notification)
- Idempotent behaviour - no duplicate approval cards on retrigger
- AI summary generation - successful call and correct Dataverse write
- AI summary fallback - model unavailable, fallback returned, request not blocked
- Overdue pending request detection by SLA watcher
- Reminder behaviour - correct ReminderCount increment and timestamp update
- Escalation trigger after reminder limit reached
- Authenticated secure variant - submission using signed-in Entra identity

13 WHAT THIS DEMONSTRATES

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

Skill Area	How It Is Demonstrated
Solution architecture	Designed a decoupled, modular system before writing a single flow - data model, lifecycle states, and component responsibilities defined first.
Dataverse as system of record	Used Dataverse to hold authoritative state and audit history, not just as a data store behind a canvas app.
Power Automate at scale	Six focused flows with clear separation of concerns, Try/Catch error handling, idempotency controls, and child flow reuse.
Copilot Studio + governance	Structured intake, policy gating, status queries, and two authentication modes in a single agent.
AI - enterprise pattern	BYOM using Azure OpenAI - config via environment variables, key in Key Vault, fail-safe fallback.
Autonomous operations	Scheduled watcher that maintains workflow health without human intervention - SLA reminder and escalation.
RBAC and security	Least-privilege roles, state-based edit controls, and field-level governance in the data model.
Entra ID authentication	Trusted identity capture in the secure variant - from signed-in context, not user-typed data.
ALM discipline	Solution-based deployment, environment variables, connection references, managed/unmanaged separation.
Operational mindset	Error capture back to Dataverse, correlation ID tracing, support-friendly design - built for production.

14 WHAT I WOULD BUILD NEXT

Scope was drawn deliberately. These are the natural next iterations for a production deployment:

- Microsoft Graph integration to validate approver email against Entra and auto-resolve group membership
- Power Apps canvas form for requestors who prefer a structured web form over a conversational interface
- Approval delegation - allowing approvers to reassign to a deputy when out of office
- Full Managed Environment deployment with DLP policies and CoE dashboard integration
- Power BI reporting layer on Dataverse for request volume, SLA compliance, and approval turnaround analytics
- Webhook to a real provisioning system - calling Microsoft Graph to assign Entra group membership on approval