**1. Azure ↔ AWS Feature Parity (Key Services to Translate)**

| **AWS Service (your original plan)** | **Azure Equivalent** |
| --- | --- |
| S3 (object storage) | **Azure Blob Storage** |
| Lambda (serverless compute) | **Azure Functions** |
| API Gateway | **Azure API Management** or **Azure Functions HTTP Trigger + Azure Front Door** |
| Textract (OCR) | **Azure Form Recognizer / Cognitive Services OCR** |
| Bedrock (LLM hosting) | **Azure OpenAI**, **Azure Cognitive Services for Language**, or **Azure Machine Learning with hosted model** |
| DynamoDB | **Azure Cosmos DB** (NoSQL) or **Azure SQL / Cosmos Table API** |
| CloudWatch / Logs | **Azure Monitor / Application Insights** |
| IAM / KMS | **Azure Active Directory / Azure Key Vault / RBAC** |

**Proposal & Stack Flexibility**

* Build a **cloud-agnostic PV prototype** that can run either on AWS or Azure.
* Use **Azure Blob Storage + Functions + Form Recognizer + Azure OpenAI** as equivalents if client prefers staying on Azure.
* Focus on **core extraction pipeline**: SharePoint → Blob Storage → OCR → LLM extraction → JSON / AE summary → audit logs.
* Output formats: JSON, AE CSV (SDTM-style), and rule-based validation + confidence scoring.
* UI layer: Simple API + web UI (e.g. React or static site) that works with either backend.
* Demonstrate both cloud paths or pick Azure for MVP to align with client’s existing environment.

**3. Updated Milestones (Azure-aware)**

* **Week 1 (Foundation):** Setup Blob Storage, Azure Functions trigger, Form Recognizer integration
* **Week 2 (Extraction):** Prompt engineering + LLM extraction (Azure OpenAI) + validation
* **Week 3 (UI & Demo):** API + web UI, logging via Application Insights, show formal demo

**4. Adjusted Risk Table (Cloud choice risk added)**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Wrong cloud choice | Wasted effort | Build abstraction ensuring minimal coupling to one platform |
| Vendor limitations (Azure LLM) | Model may not perform as well | Keep fallback path (AWS or multi-LLM strategy) |
| Integration complexity | Slower prototype | Begin with basic samples, defer edge cases |