**Artificial Intelligence- Solution Design Document**

**Author:- Gopal Soni**

**Date:- 10/05/25**

**Document Version:- 1.0**

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **S.no.** | **Title** | **Page No.** |
| 1 | Customer Use Case & Project Scope | 4 |
| 2 | User Roles & Responsibilities | 4 |
| 3 | Data Source & File Types | 4 |
| 4 | Ingestion Engine & Pipeline | 5 |
| 5 | Data Storage & Structure | 5 |
| 6 | Encryption Methodology | 6 |
| 7 | Data Preparation & Processing Layer | 7 |
| 8 | Responsible AI Standard Alignment | 7 |
| 9 | Impact Assessment Summary | 8 |
| 10 | Security Architecture | 8 |
| 11 | Sizing & Performance Planning | 9 |
| 12 | Monitoring & Diagnostics | 10 |
| 13 | Cost Optimization | 10 |
| 14 | AI Model/Service Selection | 11 |
| 15 | Inferencing & Deployment | 12 |
| 16 | Automation | 12 |
| 17 | Azure AI Architecture Best Practices | 12 |
| 18 | Physical Architecture Diagram | 13 |
| 19 | Capability Domain – Intelligent Application  Solutions | 13 |

1. **Customer Use Case & Project Scope**

Privet Venture has designed a product which is a smart document management system that helps manage documented information effortlessly**.** It is tech serving provider offering document management and automation services to law firms and corporate legal departments. These include NDAs, employment contracts, MSAs, and vendor agreements. The manual clause review process (3–5 days per document) created bottlenecks, compliance risks, and operational inefficiencies.

This solution introduces an AI-powered clause extraction platform, built on Azure, which automates identification and tagging of clauses (e.g., Confidentiality, Termination) and related metadata (e.g., contract parties, effective dates). The platform is scalable, secure, and optimized for RBAC and audit compliance.

**2.User Roles & Responsibilities**

|  |  |
| --- | --- |
| **Role** | **Description** |
| Azure Solution Architect | Overall architecture design and resource planning |
| AI Engineer | Integrated Computer Vision API and clause rule logic |
| Deployment Specialist | Provisioned Function App, Blob, Key Vault, networking |
| Security Analyst | Configured RBAC, Key Vault, and NSG/VNet settings |
| Legal Reviewer | Validated clause output accuracy and UAT testing |

**3.Data Source & File Types**

|  |  |
| --- | --- |
| **Data Source** | **File Types Ingested** |
| Secure Upload Portal | Scanned Legal Contracts (PDF, TIFF) |
| Client Legal Department | Uploaded via Web UI |
| Example Contracts | NDA, MSA, SLAs, Employment Letters |

1. **Ingestion Engine & Pipeline**

No external ingestion engine (e.g., ADF) is used due to direct upload via portal.

|  |  |
| --- | --- |
| **Component** | **Description** |
| Azure Blob Storage | Ingests and stores raw scanned documents |
| Azure Function App | Triggered by Blob upload event |

1. **Data Storage & Structure**

|  |  |  |
| --- | --- | --- |
| **Storage Layer** | **Technology Used** | **Description** |
| File Storage | Azure Blob Storage | Raw PDFs + OCR Output |
| Metadata Storage | SQL on Azure VM | Clause metadata (JSON to table) |

**6.Encryption Methodology**

|  |  |
| --- | --- |
| **Encryption Area** | **Approach** |
| Storage (Blob, SQL) | Azure Storage Encryption (AES-256) |
| Credentials/Keys | Azure Key Vault (Secret rotation every 90 days) |
| Network Communications | TLS 1.2 for all API and data transfers |

**7.Data Preparation & Processing Layer**

|  |  |
| --- | --- |
| **Stage** | **Description** |
| OCR | Azure Computer Vision Read API (v3.2) extracts text |
| Clause Tagging Logic | Rule-based logic engine running on Azure VM |
| Metadata Conversion | JSON output parsed and stored as structured metadata |

**Clause Rule Logic Overview**

Overview:

The clause tagging engine runs on an Azure VM as a lightweight script-based logic layer. It processes the OCR-extracted text using keyword and pattern-matching rules. These rules identify clause types such as Termination, Confidentiality, Force Majeure, etc.

# Logic Highlights:

* 1. Clause patterns are defined using regular expressions and legal keywords.
  2. Text blocks are segmented based on headers, bulleting, and positioning.
  3. Each clause is assigned a **confidence score** based on keyword density and positional relevance.
  4. JSON output from this logic is parsed and stored in SQL for downstream search

**8.Responsible AI Standard Alignment**

|  |  |
| --- | --- |
| **RAI Principle** | **Compliance Action** |
| Transparency | Logs retained for every OCR and clause output |
| Accountability | Logs linked to Entra ID roles for traceability |
| Fairness | No demographic data used; all documents treated equally |
| Security | Network isolation (VNet + NSG); RBAC enforced |

**9.Impact Assessment Summary**

* 1. Assessment conducted using Microsoft’s RAI Template
  2. Business improvement of 70% in manual effort
  3. Accuracy > 95% on clause tagging in environment

**Completed RAI Template**

**10.Security Architecture**

|  |  |
| --- | --- |
| **Security Control** | **Implementation Details** |
| Network Isolation | Azure VNet with NSGs blocking public access |
| Credential Protection | Azure Key Vault with RBAC policies |
| RBAC Roles | Entra ID: Admin, Reviewer, Auditor |
| Data Retention & Logging | 12-month log policy with Azure Monitor & Log Analytics |

**11.Sizing & Performance Planning**

|  |  |
| --- | --- |
| **Area** | **Metric** |
| Daily Ingestion | Up to 500 legal contracts/day |
| OCR Latency | Avg. 18–22 seconds per document |
| Function Trigger SLA | Under 5 seconds |
| Clause Accuracy | > 95% average |

**12.Monitoring & Diagnostics**

|  |  |
| --- | --- |
| **Tool** | **Monitored Metrics** |
| Azure Monitor | Function status, trigger health, error logs |
| Application Insights | OCR latency, failure rate, response tracking |
| Key Vault Logging | Secret usage, expiration tracking |

1. **Cost Optimization**

|  |  |
| --- | --- |
| **Service Area** | **Cost Strategy** |
| OCR API Usage | Pay-per-use ($1.50/1,000 pages) |
| Storage | Lifecycle management for blob containers |
| Compute (Function) | Serverless plan (Consumption model) |
| SQL VM Optimization | Auto-shutdown and off-peak scheduling configured |

**14.AI Model/Service Selection**

|  |  |
| --- | --- |
| **AI Model / API Used** | **Justification** |
| Azure Computer Vision (Read) | High OCR accuracy, cost-effective for PDF inputs |
| Rule-Based Logic Engine | No training data needed; predictable legal formats |

*No custom ML model used due to standardization of legal documents.*

**15.Inferencing & Deployment**

| Inferencing Mode | Trigger-Based OCR (Blob Storage) |

| Deployment Mode | Azure Function + API Call (No container/AKS needed) |

| Output Interaction | Stored output searchable by client application |

1. **Automation**

|  |  |
| --- | --- |
| **Activity** | **Tool or Approach** |
| Code Deployment | Via Terraform |
| Secrets Management | Azure Key Vault with role enforcement |
| Monitoring Setup | App Insights + Alerts configured |

# Function App Deployment Process Deployment Approach:

* + The Function App is deployed via Terraform under the Consumption Plan for auto- scaling.
  + Code is uploaded via ZIP package or directly published from Visual Studio Code.
  + App settings include:
    - Computer Vision API Key and Endpoint (stored in Azure Key Vault)
    - Blob Trigger settings (auto-detect new documents)
  + Monitoring is enabled via Application Insights and Log Analytics.

# Future Scope:

Pipeline will be transitioned to automated deployment using GitHub Actions or Azure DevOps for CI/CD integration.

**17.Azure AI Architecture Best Practices**

* + Serverless compute design for high scalability
  + Blob-based ingestion for seamless OCR trigger
  + Pretrained Azure Vision API to reduce training costs
  + Modular logic layer allows custom tagging rules
  + All services deployed within secured VNet

**18.Physical Architecture Diagram**

**19.Capability Domain – Intelligent Application Solutions**

|  |  |
| --- | --- |
| **Requirement** | **How It Was Addressed** |
| AI Service Pricing & Transactions | Read API pricing and scale factored into solution |
| Interaction & App Layer Design | Client UI integrates output via SQL-layer downstream |
| Container/Native App Design | Serverless functions used; no containerization required |

**END OF DOCUMENT**