Ottawa GenAI Research Assistant

Sprint Planning Document

Project: Ottawa Economic Development GenAI Research Assistant
Version: 1.0
Date: February 4, 2026
Sprint Duration: 2 weeks per sprint
Total Duration: 12 weeks (6 sprints)

Team Members

Name	Role	Responsibilities
Travis Yi	Tech Lead / Full Stack Developer	Architecture design, backend development, Azure integration
Peng Wang	Backend Developer	RAG implementation, API development, LLM integration
Hye Ran Yoo	Frontend Developer / QA	Frontend development, UI/UX, testing, documentation

---

Executive Summary

This document outlines the complete sprint planning for the Ottawa GenAI Research Assistant project. The project aims to build an AI-powered research assistant that helps economic development analysts query quarterly reports using natural language, with citation-backed responses and visualization capabilities.

Project Goals

- Build a RAG-based intelligent assistant for economic development analysts
- Implement natural language Q&A without system redeployment
- Provide trusted outputs with source citations and confidence indicators
- Support English and French bilingual interaction
- Deploy on fully Azure cloud-native architecture

Key Metrics

Metric	Target
Answer Accuracy	≥75%
Faithfulness	≥90%
Context Recall	≥85%
Response Time (P95)	<3s
System Availability	≥99.5%

Sprint Roadmap Overview

Sprint	Duration	Phase	Primary Goal	Story Points
Sprint 1	Feb 9 - Feb 20	Phase 1	Azure Infrastructure Setup	21
Sprint 2	Feb 23 - Mar 6	Phase 1	Frontend Migration & Authentication	18
Sprint 3	Mar 9 - Mar 20	Phase 2	RAG Core Implementation	23
Sprint 4	Mar 23 - Apr 3	Phase 2	Search & Chat Features	19
Sprint 5	Apr 6 - Apr 17	Phase 3	Advanced Features & Visualization	20
Sprint 6	Apr 20 - May 1	Phase 3	Production Deployment & Documentation	17

Total Story Points: 118

Phase 1: Infrastructure Migration (Sprints 1-2)

Objective

Migrate the project from local/hybrid architecture to a fully Azure cloud-native architecture.

Sprint 1: Azure Infrastructure Setup

Duration: February 9 - February 20, 2026
Sprint Goal: Establish core Azure cloud infrastructure
Capacity: 21 Story Points

Sprint Backlog

🔴 **High Priority (P0 - Must Have)**

US-102: Azure Storage Configuration

Story Points: 5 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As a*** *system administrator,*
> ***I want to*** *configure Azure Blob Storage as document storage,*
> ***So that*** *documents can be securely and persistently stored.*

Acceptance Criteria:

- [ ] Create Azure Blob Storage account (Standard LRS)
- [ ] Configure container access policies
- [ ] Store connection string in Azure Key Vault
- [ ] Backend can access Blob Storage via SDK
- [ ] Support PDF file upload and download

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create Azure Blob Storage account	2h	Travis Yi	
2	Configure container and access policies	2h	Travis Yi	
3	Set up Azure Key Vault	3h	Travis Yi	
4	Implement backend Blob Storage SDK integration	6h	Peng Wang	
5	Create upload/download API endpoints	4h	Peng Wang	
6	Write unit tests for storage operations	3h	Hye Ran Yoo	

Dependencies: None

US-103: Azure AI Search Configuration

Story Points: 8 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As a*** *system administrator,*
> ***I want to*** *configure Azure AI Search as vector storage,*
> ***So that*** *it replaces local vector storage (FAISS/Chroma).*

Acceptance Criteria:

- [ ] Create Azure AI Search service (Standard S1)
- [ ] Create document index (with text and vector fields)
- [ ] Configure hybrid search (Vector + Keyword)
- [ ] Store API Key in Azure Key Vault
- [ ] Backend can execute search queries
- [ ] **Remove** all local vector storage code

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create Azure AI Search service	2h	Travis Yi	
2	Design and create document index schema	4h	Travis Yi	
3	Configure hybrid search settings	3h	Peng Wang	
4	Implement search SDK integration	8h	Peng Wang	
5	Remove FAISS/Chroma local storage code	4h	Peng Wang	
6	Test hybrid search functionality	4h	Hye Ran Yoo	

Dependencies: US-102 (Key Vault)

US-104: Azure OpenAI Configuration

Story Points: 5 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As a*** *system administrator,*
> ***I want to*** *configure Azure OpenAI / AI Foundry service,*
> ***So that*** *I can use enterprise-grade LLM endpoints.*

Acceptance Criteria:

- [ ] Deploy GPT-4o model
- [ ] Deploy text-embedding-ada-002 model
- [ ] Store endpoint and key in Azure Key Vault
- [ ] Backend calls via Azure OpenAI SDK
- [ ] **Remove** all direct OpenAI endpoint code

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create Azure AI Foundry resource	2h	Travis Yi	
2	Deploy GPT-4o model	2h	Travis Yi	
3	Deploy embedding model	2h	Travis Yi	
4	Implement Azure OpenAI SDK integration	6h	Peng Wang	
5	Remove direct OpenAI API code	3h	Peng Wang	
6	Test LLM API calls	3h	Hye Ran Yoo	

Dependencies: US-102 (Key Vault)

🟡 **Medium Priority (P1 - Should Have)**

Backend Azure SDK Integration

Story Points: 3 | **Priority**: P1 | **Status**: To Do

Description: Integrate all Azure SDKs into the FastAPI backend with proper configuration management.

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create Azure configuration module	4h	Peng Wang	
2	Implement environment variable management	3h	Peng Wang	
3	Create health check endpoints	2h	Peng Wang	

Dependencies: US-102, US-103, US-104

Sprint 1 Definition of Done ✅

- [ ] All Azure services provisioned and accessible
- [ ] Connection strings and API keys stored in Key Vault

- [ ] Backend can connect to all Azure services
- [ ] Local vector storage code removed
- [ ] All tests passing
- [ ] Documentation updated

Sprint 2: Frontend Migration & Authentication

Duration: February 23 - March 6, 2026
Sprint Goal: Complete frontend migration to Vite and implement Azure Entra ID authentication
Capacity: 18 Story Points

Sprint Backlog

🔴 **High Priority (P0 - Must Have)**

US-101: Frontend Framework Migration

Story Points: 8 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As a*** *developer,*
> ***I want to*** *migrate the frontend from Create React App to Vite,*
> ***So that*** *I can eliminate dependency vulnerabilities and improve development experience.*

Acceptance Criteria:

- [ ] Initialize new project structure with Vite
- [ ] Migrate all existing components and routes
- [ ] Maintain same functionality and UI
- [ ] Build artifacts size ≤ CRA version
- [ ] Dev server hot reload works correctly
- [ ] No high-risk npm audit vulnerabilities

Task Breakdown:

#	Task	Hours	Owner	Status
1	Initialize Vite project with React + TypeScript	2h	Hye Ran Yoo	
2	Configure build settings and paths	3h	Hye Ran Yoo	
3	Migrate core components	8h	Hye Ran Yoo	
4	Migrate routing configuration	3h	Hye Ran Yoo	
5	Update API integration layer	4h	Hye Ran Yoo	
6	Run security audit and fix vulnerabilities	2h	Hye Ran Yoo	
7	Verify build artifacts and performance	2h	Travis Yi	

Dependencies: None

US-105: Azure Entra ID Authentication

Story Points: 8 | **Priority**: P0 | **Status**: 🟪 To Do

User Story:

> ***As a*** *user,*
> ***I want to*** *log in via Azure Entra ID,*
> ***So that*** *I can use enterprise single sign-on.*

Acceptance Criteria:

- ☐ Create Azure AD App Registration
- ☐ Configure Redirect URI
- ☐ Frontend integrates MSAL.js
- ☐ Backend validates JWT Token
- ☐ Unauthenticated users redirect to login
- ☐ **Remove** all Google OAuth code

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create Azure AD App Registration	2h	Travis Yi	🟪
2	Configure redirect URIs and permissions	2h	Travis Yi	🟪
3	Implement MSAL.js in frontend	6h	Hye Ran Yoo	🟪
4	Create authentication context/provider	4h	Hye Ran Yoo	🟪
5	Implement backend JWT validation	5h	Peng Wang	🟪
6	Create protected route middleware	3h	Peng Wang	🟪
7	Remove Google OAuth code	2h	Hye Ran Yoo	🟪
8	End-to-end authentication testing	3h	Hye Ran Yoo	🟪

Dependencies: None

🟡 **Medium Priority (P1 - Should Have)**

Delete Legacy Code Cleanup

Story Points: 2 | **Priority**: P1 | **Status**: 🟪 To Do

Description: Remove all deprecated local storage and authentication code.

Task Breakdown:

#	Task	Hours	Owner	Status
1	Remove CRA configuration files	1h	Hye Ran Yoo	🟪
2	Clean up deprecated dependencies	2h	Hye Ran Yoo	🟪
3	Update documentation	2h	Hye Ran Yoo	🟪

Dependencies: US-101, US-105

Sprint 2 Definition of Done ✅

- [ ] Frontend runs on Vite with no vulnerabilities
- [ ] Users can authenticate via Azure Entra ID
- [ ] Protected routes working correctly
- [ ] All Google OAuth and CRA code removed
- [ ] All tests passing
- [ ] Documentation updated

Phase 2: Core RAG Functionality (Sprints 3-4)

Objective

Implement end-to-end RAG (Retrieval-Augmented Generation) functionality, including automatic document pipeline, intelligent retrieval, and citation generation.

Sprint 3: RAG Core Implementation

Duration: March 9 - March 20, 2026
Sprint Goal: Implement document ingestion pipeline and RAG orchestrator
Capacity: 23 Story Points

Sprint Backlog

🔴 **High Priority (P0 - Must Have)**

US-201: Automatic Document Pipeline

Story Points: 13 | **Priority**: P0 | **Status**: ⬜ To Do

User Story:

> ***As a*** *system administrator,*
> ***I want to*** *documents to automatically sync from the economic portal to the RAG system,*
> ***So that*** *data stays current without manual uploads.*

Acceptance Criteria:

- [ ] Documents stored in Azure Blob Storage / Microsoft Fabric
- [ ] New portal documents automatically trigger processing pipeline
- [ ] Pipeline executes: Chunking → Embedding Generation → Index to Azure AI Search
- [ ] RAG Pipeline Mapping configured from Azure Portal
- [ ] Support viewing processing status (Pending → Processing → Indexed)

Task Breakdown:

#	Task	Hours	Owner	Status
1	Design document processing architecture	4h	Travis Yi	⬜

2	Implement PDF text extraction	6h	Peng Wang	
3	Implement document chunking logic	6h	Peng Wang	
4	Create embedding generation service	5h	Peng Wang	
5	Implement Azure AI Search indexing	6h	Peng Wang	
6	Create Azure Function trigger for new documents	4h	Travis Yi	
7	Build document status tracking API	4h	Peng Wang	
8	Create document status UI component	4h	Hye Ran Yoo	
9	End-to-end pipeline testing	4h	Hye Ran Yoo	

Dependencies: Sprint 1, Sprint 2

RAG Orchestrator Implementation

Story Points: 8 | **Priority**: P0 | **Status**: To Do

Description: Core RAG orchestration logic that coordinates retrieval and generation.

Technical Specifications:

Parameter	Specification
Embedding Model	text-embedding-ada-002
Generation Model	GPT-4o (Azure AI Foundry)
Retrieval Strategy	Hybrid Search (Vector + Keyword)
Top-K	5 chunks
Ranking Algorithm	Cosine Similarity + BM25
Fallback Strategy	Return "No relevant information found" when no match

Task Breakdown:

#	Task	Hours	Owner	Status
1	Design RAG orchestrator architecture	3h	Travis Yi	
2	Implement query preprocessing	4h	Peng Wang	
3	Implement hybrid search integration	6h	Peng Wang	
4	Create response generation with context	6h	Peng Wang	
5	Implement prompt templates	4h	Peng Wang	
6	Unit testing for orchestrator	4h	Hye Ran Yoo	

Dependencies: US-201

🟡 **Medium Priority (P1 - Should Have)**

Prompt Optimization

Story Points: 2 | **Priority**: P1 | **Status**: ⬜ To Do

Task Breakdown:

#	Task	Hours	Owner	Status
1	Design system prompts for accuracy	3h	Peng Wang	⬜
2	Implement prompt versioning	2h	Peng Wang	⬜
3	Create prompt testing framework	3h	Peng Wang	⬜

Dependencies: RAG Orchestrator

Sprint 3 Definition of Done ✅

- ☐ Documents can be uploaded and automatically processed
- ☐ Chunking and embedding generation working
- ☐ Documents indexed in Azure AI Search
- ☐ RAG orchestrator returns contextual responses
- ☐ All tests passing
- ☐ Documentation updated

Sprint 4: Search & Chat Features

Duration: March 23 - April 3, 2026
Sprint Goal: Implement natural language search with citations and chat history
Capacity: 19 Story Points

Sprint Backlog

🔴 **High Priority (P0 - Must Have)**

US-202: Natural Language Query

Story Points: 5 | **Priority**: P0 | **Status**: ⬜ To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want to*** *ask questions in natural language,*
> ***So that*** *I can quickly get relevant quarterly report data.*

Acceptance Criteria:

- ☐ User input returns results within 3 seconds
- ☐ Retrieval uses Top-5 hybrid search (Cosine + BM25)
- ☐ Results sorted by relevance
- ☐ Support English and French queries

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create chat input component	3h	Hye Ran Yoo	
2	Implement query API endpoint	4h	Peng Wang	
3	Optimize response time	3h	Peng Wang	
4	Implement relevance sorting	3h	Peng Wang	
5	Add loading states and error handling	2h	Hye Ran Yoo	

Dependencies: Sprint 3

US-203: Citation-Backed Response Generation

Story Points: 5 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want*** *responses to include specific source citations,*
> ***So that*** *I can verify answer credibility.*

Acceptance Criteria:

- [ ] Each response includes at least one source citation
- [ ] Citation format: [Document Name, Page Number, Original Text Excerpt]
- [ ] Display confidence indicator (High/Medium/Low)
- [ ] Clicking citation opens original text preview
- [ ] Display "No relevant information found" when content unavailable

Task Breakdown:

#	Task	Hours	Owner	Status
1	Implement citation extraction logic	5h	Peng Wang	
2	Create citation formatting service	3h	Peng Wang	
3	Build citation UI component	4h	Hye Ran Yoo	
4	Implement document preview modal	4h	Hye Ran Yoo	
5	Add confidence indicator display	2h	Hye Ran Yoo	

Dependencies: US-202

US-204: Chat History Persistence

Story Points: 5 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want*** *chat history to be saved,*
> ***So that*** *I can continue previous conversations on next login.*

Acceptance Criteria:

- [ ] Chat history stored per user in Cosmos DB
- [ ] Support viewing history conversation list
- [ ] Support deleting history records
- [ ] Support continuing historical conversations

Task Breakdown:

#	Task	Hours	Owner	Status
1	Set up Azure Cosmos DB	2h	Travis Yi	
2	Design chat history data model	2h	Peng Wang	
3	Implement chat history API	5h	Peng Wang	
4	Create chat history sidebar UI	4h	Hye Ran Yoo	
5	Implement delete and continue functions	3h	Hye Ran Yoo	

Dependencies: US-202

🟡 **Medium Priority (P1 - Should Have)**

US-205: Bilingual Support (i18n)

Story Points: 4 | **Priority**: P1 | **Status**: To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want to*** *switch between English and French interfaces,*
> ***So that*** *I can share results with French-speaking users.*

Acceptance Criteria:

- [ ] Interface supports one-click English/French switching
- [ ] All UI text managed via i18n
- [ ] Query and response support bilingual processing
- [ ] Language preference persisted

Task Breakdown:

#	Task	Hours	Owner	Status
1	Set up i18n framework (react-i18next)	3h	Hye Ran Yoo	
2	Create English translation file	4h	Hye Ran Yoo	
3	Create French translation file	4h	Hye Ran Yoo	
4	Implement language switcher component	2h	Hye Ran Yoo	
5	Persist language preference	1h	Hye Ran Yoo	

Dependencies: None

Sprint 4 Definition of Done ✅

- [ ] Natural language queries return relevant results
- [ ] Responses include proper citations
- [ ] Chat history persisted to Cosmos DB
- [ ] Bilingual UI switching works
- [ ] All tests passing
- [ ] Documentation updated

Phase 3: Advanced Features & Production (Sprints 5-6)

Objective

Implement advanced analytics features, quality evaluation framework, and complete production deployment.

Sprint 5: Advanced Features & Visualization

Duration: April 6 - April 17, 2026
Sprint Goal: Implement data visualization and LLM evaluation framework
Capacity: 20 Story Points

Sprint Backlog

🔴 **High Priority (P0 - Must Have)**

US-301: Chart Visualization

Story Points: 8 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want*** *chat responses to include data charts,*
> ***So that*** *I can intuitively understand economic trends.*

Acceptance Criteria:

- [ ] Support line charts, bar charts, pie charts
- [ ] Charts dynamically generated from retrieved data
- [ ] Support client-side rendering (Recharts)
- [ ] Charts exportable as PNG/PDF
- [ ] Charts include data source annotation

Task Breakdown:

#	Task	Hours	Owner	Status
1	Set up Recharts library	2h	Hye Ran Yoo	
2	Create chart components (line, bar, pie)	8h	Hye Ran Yoo	
3	Implement data extraction for charts	6h	Peng Wang	

4	Create chart generation API	4h	Peng Wang	
5	Implement chart export functionality	4h	Hye Ran Yoo	
6	Add source annotations	2h	Hye Ran Yoo	

Dependencies: Sprint 4

US-302: Dynamic Report Dashboard

Story Points: 5 | **Priority**: P0 | **Status**: To Do

User Story:

> ***As an*** *economic development analyst,*
> ***I want*** *the reports page to display real statistics,*
> ***So that*** *I can understand system usage.*

Acceptance Criteria:

- [ ] Report data from backend API (not hardcoded)
- [ ] Display total documents, indexed count, query count
- [ ] Support filtering by time range
- [ ] Data updates in real-time

Task Breakdown:

#	Task	Hours	Owner	Status
1	Design statistics data model	2h	Peng Wang	
2	Create statistics API endpoints	5h	Peng Wang	
3	Build dashboard UI components	6h	Hye Ran Yoo	
4	Implement time range filtering	3h	Hye Ran Yoo	
5	Add real-time updates	2h	Hye Ran Yoo	

Dependencies: Sprint 4

🟡 **Medium Priority (P1 - Should Have)**

US-303: LLM Evaluation Framework

Story Points: 7 | **Priority**: P1 | **Status**: To Do

User Story:

> ***As a*** *developer,*
> ***I want to*** *automatically evaluate LLM response quality,*
> ***So that*** *I can continuously monitor and optimize the system.*

Acceptance Criteria:

- [ ] Implement 6-dimension evaluation: Coherence, Relevancy, Completeness, Grounding, Helpfulness, Faithfulness

- [ ] Each dimension scored 1-5
- [ ] Evaluation results stored and queryable
- [ ] Alerts triggered when below threshold
- [ ] Generate periodic evaluation reports

Evaluation Specifications:

Dimension	Description	Target Score
Coherence	Response logical coherence	≥ 4.0/5.0
Relevancy	Response relevance to question	≥ 4.0/5.0
Completeness	Response completeness	≥ 3.5/5.0
Grounding	Response based on retrieved content (no hallucination)	≥ 4.5/5.0
Helpfulness	Response usefulness	≥ 4.0/5.0
Faithfulness	Response consistency with citations	≥ 4.5/5.0

Task Breakdown:

#	Task	Hours	Owner	Status
1	Design evaluation service architecture	3h	Travis Yi	
2	Implement 6-dimension evaluation logic	8h	Peng Wang	
3	Create evaluation results storage	3h	Peng Wang	
4	Build evaluation dashboard UI	5h	Hye Ran Yoo	
5	Set up alerts for low scores	2h	Travis Yi	

Dependencies: Sprint 4

Sprint 5 Definition of Done ✅

- [ ] Chart visualization working in chat responses
- [ ] Dashboard displays real-time statistics
- [ ] LLM evaluation framework operational
- [ ] All tests passing
- [ ] Documentation updated

Sprint 6: Production Deployment & Documentation

Duration: April 20 - May 1, 2026
Sprint Goal: Deploy to production and complete all documentation
Capacity: 17 Story Points

Sprint Backlog

🔴 High Priority (P0 - Must Have)

US-304: Production Deployment

Story Points: 8 | **Priority**: P0 | **Status**: 🟪 To Do

User Story:

> ***As a*** *system administrator,*
> ***I want to*** *deploy the system to Azure production environment,*
> ***So that*** *stable service is provided.*

Acceptance Criteria:

- [ ] Frontend deployed to Azure Static Web Apps
- [ ] Backend deployed to Azure Container Apps
- [ ] Application Insights monitoring configured
- [ ] Alert rules configured
- [ ] System availability ≥ 99.5%

Task Breakdown:

#	Task	Hours	Owner	Status
1	Configure Azure Static Web Apps	3h	Travis Yi	🟪
2	Configure Azure Container Apps	4h	Travis Yi	🟪
3	Set up Application Insights	3h	Travis Yi	🟪
4	Configure CI/CD pipelines	5h	Travis Yi	🟪
5	Set up monitoring alerts	2h	Travis Yi	🟪
6	Perform load testing	4h	Hye Ran Yoo	🟪
7	Production smoke testing	3h	Hye Ran Yoo	🟪

Dependencies: Sprint 5

US-305: Demo Documentation

Story Points: 5 | **Priority**: P0 | **Status**: 🟪 To Do

User Story:

> ***As a*** *project manager,*
> ***I want*** *complete demo materials,*
> ***So that*** *I can showcase results to stakeholders.*

Acceptance Criteria:

- [ ] Demo video (5-10 minutes)
- [ ] User guide
- [ ] API documentation
- [ ] Azure resource configuration screenshots

- [ ] Live API call demo

Task Breakdown:

#	Task	Hours	Owner	Status
1	Create demo script and storyboard	2h	Hye Ran Yoo	
2	Record demo video	4h	Hye Ran Yoo	
3	Write user guide	6h	Hye Ran Yoo	
4	Generate API documentation	4h	Peng Wang	
5	Capture Azure resource screenshots	2h	Travis Yi	
6	Prepare live demo environment	2h	Travis Yi	

Dependencies: US-304

🟡 **Medium Priority (P1 - Should Have)**

CI/CD Pipeline

Story Points: 4 | **Priority**: P1 | **Status**: To Do

Task Breakdown:

#	Task	Hours	Owner	Status
1	Configure GitHub Actions for frontend	3h	Travis Yi	
2	Configure GitHub Actions for backend	3h	Travis Yi	
3	Set up staging environment	3h	Travis Yi	
4	Implement automated testing in pipeline	3h	Travis Yi	

Dependencies: US-304

Sprint 6 Definition of Done ✅

- [ ] Application deployed to Azure production
- [ ] Monitoring and alerting operational
- [ ] CI/CD pipelines working
- [ ] Demo video completed
- [ ] All documentation delivered
- [ ] Project handoff completed

Milestones

Milestone	Target Date	Phase	Deliverables
Phase 1 Complete	March 6, 2026	Phase 1	Azure infrastructure, Vite migration, Authentication

Phase 2 Complete	April 3, 2026	Phase 2	RAG functionality, Natural language search, Chat history
Phase 3 Complete	May 1, 2026	Phase 3	Visualization, LLM evaluation, Production deployment
Project Complete	May 1, 2026	-	Full system delivery with documentation

---

Risk Management

Risk	Impact	Probability	Mitigation
Vite migration compatibility issues	Medium	Low	Gradual migration, keep CRA backup
Azure service quota limitations	High	Medium	Request quota increase in advance
Authentication integration complexity	Medium	Medium	Use official MSAL examples
RAG quality instability	High	Medium	Iterative prompt optimization
Pipeline trigger delays	Medium	Low	Use Azure Functions event trigger
Bilingual translation quality	Medium	Low	Professional translation service review
Chart generation latency	Low	Low	Client-side rendering to reduce backend load
Evaluation cost high	Medium	Medium	Sampling evaluation instead of full evaluation
Deployment complexity	Medium	Low	Use Infrastructure as Code

Success Criteria

Technical Criteria

- [ ] Frontend uses Vite build with no vulnerabilities
- [ ] All storage uses Azure services
- [ ] All secrets stored in Key Vault
- [ ] Users can authenticate via Entra ID
- [ ] No local vector storage code remains
- [ ] Documents auto-sync from portal and indexed
- [ ] Users can query with natural language and get cited responses
- [ ] Chat history persisted to Cosmos DB
- [ ] Interface supports English/French switching
- [ ] Users receive chart visualizations in responses
- [ ] Report dashboard shows real-time data
- [ ] LLM evaluation framework runs automatically
- [ ] System deployed to Azure production

KPI Targets

Metric	Target
Answer Accuracy	≥75%
Faithfulness	≥90%
Context Recall	≥85%
Response Time (P95)	<3s
Search Latency	<500ms
User Satisfaction	>4.0/5.0
System Availability	≥99.5%
LLM Evaluation Score	≥4.0/5.0

Appendix

A. Technology Stack

Layer	Technology
Frontend	React 18 + TypeScript (Vite)
Backend	FastAPI 0.104+ (Python 3.12)
Vector Store	Azure AI Search
Embedding	Azure OpenAI (text-embedding-ada-002)
LLM	Azure AI Foundry (GPT-4o)
Storage	Azure Blob Storage / Microsoft Fabric
Chat History	Azure Cosmos DB
Authentication	Azure Entra ID
Key Management	Azure Key Vault
Monitoring	Azure Application Insights

B. Budget Constraints

- **Monthly Azure Spending**: $500 - $1000
- **Data Region**: Canada (if required)
- **Data Scope**: Public documents only

C. Document References

- Master PRD
- Phase 1 PRD
- Phase 2 PRD
- Phase 3 PRD

Document Prepared By: Ottawa GenAI Research Assistant Team
Last Updated: February 4, 2026
Next Review: February 10, 2026 (Sprint 1 Planning)