## **Statement of Work (SoW) for FullStack Agentic AI Application**

### **1. Project Overview**

The purpose of this project is to design, develop, and deliver a full-stack, multi-tenant application utilizing **Agentic AI** for intelligent processing. The application will feature a **ReactJS frontend**, API integrations with MongoDB and upstream/downstream systems, AI-driven QA workflows powered by **LangGraph or Google SDK**, will validate from partner systems for reference data and a real-time dashboard.**2. Scope of Work**

#### **2.1 Frontend (ReactJS)**

* Develop a responsive, intuitive UI using **ReactJS**.
* Implement authentication (OAuth2, JWT) with tenant-level isolation.
* Role-based access control (Admin, Manager, Analyst, Viewer).
* UI Components:
  + Upload/Trigger Input Interface
  + QA Workflow Status and Logs
  + Dashboard with Visualizations
  + History and Audit Log Viewer
  + Settings & Tenant Configuration

#### **2.2 Backend (NodeJS/RestAPI JavaSprintboot)**

* Build RESTful to:  
  + Connect to upstream MongoDB (source)
  + Push results to downstream systems
  + Handle tenant-specific logic

#### **2.3 Agentic AI Workflow**

* Framework: Use **LangGraph** or **Google SDK** to develop an intelligent **QA Agent** pipeline.
* Capabilities:  
  + Accept structured/unstructured data
  + Perform validation against different sysyem, enrichment, summarization, and classification tasks
  + Chain multiple agents for context-aware decisions
  + Output results for persistence

#### **2.4 Database Integration**

* **MongoDB (Upstream & Downstream):**
  + Read from a designated source collection (per tenant)
  + Write processed output to a different destination collection
  + Store logs, agent decisions, and QA outcomes

#### **2.5 Multi-Tenancy Architecture**

* Logical separation of tenant data via:
  + Database schema design
  + AuthN/AuthZ logic
  + Config-driven agent logic per tenant

#### **2.6 Dashboard & Visualization**

* Real-time UI dashboard from MongoDB:
  + Task progress, status, success/failure rate
  + Agent performance over time
  + Filter by date, type, tenant, etc.
  + Export to CSV, PDF

#### **2.7 Downstream System Integration**

* Trigger API call or DB insert after agent processing completes.
* Configurable mapping per tenant to handle downstream schema.

#### **2.8 DevOps and Deployment**

* Containerized deployment using **Docker** and Deploy to Openshift Kubernetees.
* CI/CD pipelines (harness, Github.).
* Environment segregation (Dev, QA, Prod).

**3. Deliverables**

1. Complete FullStack Application (Frontend + Backend)
2. AI Agentic QA Workflow Implementation
3. MongoDB Integration (Read/Write)
4. Dashboards with Filtering & Exporting
5. Multi-Tenant System Architecture
6. Upstream and Downstream API Connectors
7. Documentation:
   * Codebase and Setup Instructions
   * API Documentation (Swagger/Postman)
   * User and Admin Manual
   * Tenant Configuration Guide
8. Test Cases & QA Report
9. Deployment Scripts and Infra Documentation

### **4. Timeline**

| **Phase** | **Deliverable** | **Duration** |
| --- | --- | --- |
| Discovery & Architecture | Requirements + Design Spec | 2 weeks |
| Frontend Development | React UI Implementation | 2 weeks |
| Backend APIs & Auth | MongoDB APIs, AuthN/Z | 2 weeks |
| AI Workflow | LangGraph/GoogleSDK Pipelines | 3 weeks |
| Dashboard | UI + API + Charts | 2 week |
| Integration & Testing | End-to-end testing, edge cases | 2 week |
| Deployment | Prod-ready setup | 1 week |
| Buffer | Final Fixes and QA | 1 week |

**Total Estimated Duration: 15 weeks - 8 Sprints Max**

### **5. Acceptance Criteria**

* All features are demoed with real data
* No cross-tenant data leakage
* Agentic AI pipeline outputs are traceable and explainable
* All APIs are secured and documented
* Dashboards accurately reflect data
* Successful test cases against mock upstream/downstream systems
* CI/CD pipeline passes all checks