# AI Knowledge Management System

A comprehensive, cloud-native knowledge management system built with MCP (Model Context Protocol) servers, featuring AI-powered document processing, semantic search, and knowledge graph capabilities.

## 🏗️ Architecture Overview

The system consists of multiple specialized MCP servers orchestrated through a central workflow engine:

- **Phi4 MCP Server**: AI classification and entity extraction using Phi4 model
- **Azure SQL MCP Server**: Document storage and metadata management
- **GraphRAG MCP Server**: Knowledge graph construction and relationship mapping
- **Search MCP Server**: Semantic and vector search capabilities
- **Orchestrator**: Central workflow coordination and API gateway
- **Astro Frontend**: Modern, responsive user interface

## 🌐 Azure Resources

Deployed on Azure with the following services:

| Service | Azure Resource Name | Purpose |
|---|---|---|
| Static Web App | `knowledge-system-ui` | Frontend hosting |
| App Service | `phi4-mcp-server` | AI processing server |
| App Service | `knowledge-base-sql-server` | Database operations |
| App Service | `knowledge-base-graphrag-server` | Knowledge graph operations |
| App Service | `knowledge-base-search-server` | Search operations |
| App Service | `knowledge-base-orchestrator` | Central orchestrator |
| SQL Database | `knowledge-base` | Document and metadata storage |
| SQL Server | `knowledge-sql` | Database server |
| AI Foundry | `knowledge-ai-foundry` | AI model hosting |
| Storage Account | `knowledgestorageacct` | File storage |
| App Service Plan | `mcp-app-service-plan` | Hosting plan |

## 🚀 Features

### Core Capabilities

- **AI-Powered Document Processing**: Automatic classification and content analysis
- **Entity Extraction**: Identify people, organizations, concepts, and relationships
- **Knowledge Graph**: Build and visualize connected knowledge networks
- **Semantic Search**: Advanced search using AI embeddings and natural language
- **Multi-format Support**: PDF, DOC, DOCX, TXT, and Markdown files
- **Real-time Processing**: Live status updates and progress tracking

### Advanced Features

- **Batch Processing**: Handle multiple documents simultaneously
- **Graph Visualization**: Interactive knowledge network exploration
- **AI Insights**: Generate insights and summaries from knowledge base
- **Search Analytics**: Track and optimize search patterns
- **Workflow Orchestration**: Coordinated multi-step document processing

## 📁 Repository Structure

```
knowledge-system/
├── .github/
│   └── workflows/
│       ├── deploy-frontend.yml
│       ├── deploy-mcp-servers.yml
│       ├── deploy-orchestrator.yml
│       └── test-integration.yml
├── mcp-servers/
│   ├── phi4-server/
│   │   ├── package.json
│   │   ├── server.js
│   │   └── web.config
│   ├── sql-server/
│   │   ├── package.json
│   │   ├── server.js
│   │   └── web.config
│   ├── graphrag-server/
│   │   ├── package.json
│   │   ├── server.js
│   │   └── web.config
│   └── search-server/
│       ├── package.json
│       ├── server.js
│       └── web.config
├── orchestrator/
│   ├── package.json
│   ├── server.js
│   ├── web.config
│   └── lib/
│       ├── mcpClient.js
│       └── workflowEngine.js
├── frontend/
│   ├── package.json
│   ├── astro.config.mjs
│   ├── tsconfig.json
│   ├── src/
│   │   ├── layouts/
│   │   │   └── Layout.astro
│   │   └── pages/
│   │       ├── index.astro
│   │       ├── upload.astro
│   │       └── search.astro
│   └── public/
│       └── favicon.svg
├── database/
│   ├── schema.sql
│   └── seed-data.sql
├── docker-compose.yml
├── README.md
└── .gitignore
```

## 🛠️ Setup and Deployment

### Prerequisites

- Azure subscription with appropriate permissions
- Node.js 18+ installed locally
- Git for version control
- Azure CLI (for local development)

### Azure Services Configuration

1. **Create Resource Group**:

   ```bash
   az group create --name mcp-knowledge-system --location "East US 2"
   ```

2. **Deploy Azure Resources** (use Azure Portal or ARM templates):
   - App Service Plan: `mcp-app-service-plan`
   - App Services: 5 services for MCP servers and orchestrator
   - Azure SQL Database: `knowledge-sql` server with `knowledge-base` database
   - Static Web App: `knowledge-system-ui`
   - Storage Account: `knowledgestorageacct`
   - AI Foundry: `knowledge-ai-foundry`

3. **Configure GitHub Secrets**:

   ```
   AZURE_STATIC_WEB_APPS_API_TOKEN
   AZURE_PHI4_MCP_PUBLISH_PROFILE
   AZURE_SQL_MCP_PUBLISH_PROFILE
   AZURE_GRAPHRAG_MCP_PUBLISH_PROFILE
   AZURE_SEARCH_MCP_PUBLISH_PROFILE
   AZURE_ORCHESTRATOR_PUBLISH_PROFILE
   ```

### Database Setup

1. **Connect to Azure SQL Database**
2. **Run schema creation**:

   ```sql
   -- Execute database/schema.sql
   ```

3. **Load sample data** (optional):

   ```sql
   -- Execute database/seed-data.sql
   ```

### Environment Variables

Set these in Azure App Service Configuration:

#### For SQL Server MCP:

```
AZURE_SQL_SERVER=knowledge-sql.database.windows.net
AZURE_SQL_DATABASE=knowledge-base
AZURE_SQL_USERNAME=your_username
AZURE_SQL_PASSWORD=your_password
```

#### For Search Server MCP:

```
AZURE_SEARCH_ENDPOINT=https://your-search-service.search.windows.net
AZURE_SEARCH_API_KEY=your_search_api_key
AZURE_SEARCH_INDEX=knowledge-base-index
```

#### For Orchestrator:

```
PHI4_SERVER_URL=https://phi4-mcp-server.azurewebsites.net
SQL_SERVER_URL=https://knowledge-base-sql-server.azurewebsites.net
GRAPHRAG_SERVER_URL=https://knowledge-base-graphrag-server.azurewebsites.net
SEARCH_SERVER_URL=https://knowledge-base-search-server.azurewebsites.net
```

### Local Development

1. **Clone the repository**:

   ```bash
   ```

```
git clone https://github.com/software-tim/knowledge-system.git
cd knowledge-system
```

2. **Install dependencies for each service**:

```bash
# Frontend
cd frontend && npm install

# Orchestrator
cd ../orchestrator && npm install

# Each MCP server
cd ../mcp-servers/phi4-server && npm install
cd ../sql-server && npm install
cd ../graphrag-server && npm install
cd ../search-server && npm install
```

3. **Start development servers**:

```bash
# Terminal 1: Frontend
cd frontend && npm run dev

# Terminal 2: Orchestrator
cd orchestrator && npm run dev

# Terminal 3-6: MCP Servers
cd mcp-servers/phi4-server && npm run dev
cd mcp-servers/sql-server && npm run dev
cd mcp-servers/graphrag-server && npm run dev
cd mcp-servers/search-server && npm run dev
```

## Deployment

Deployment is automated through GitHub Actions when you push to the `main` branch:

1. **Frontend**: Deployed to Azure Static Web Apps
2. **MCP Servers**: Deployed to respective Azure App Services
3. **Orchestrator**: Deployed to Azure App Service

Monitor deployments in the GitHub Actions tab of your repository.

## 📖 API Documentation

### Orchestrator Endpoints

#### POST `/api/process-document`

Process a single document with AI analysis.

**Request**:

```javascript
// Form data with file upload
{
  file: File,
  title: string,
  options: {
    enable_classification: boolean,
    enable_entities: boolean,
    enable_graph: boolean
  }
}
```

**Response**:

```javascript
{
  success: true,
  document_id: "12345",
  processing_results: {
    classification: "Technical Documentation",
    entities_extracted: 15,
    graph_relationships: 8
  }
}
```

#### POST `/api/search`

Search the knowledge base using various methods.

**Request**:

```javascript
```

```
{
  query: "machine learning",
  filters: {
    search_type: "semantic",
    content_type: "documents",
    classification: "Technical Documentation"
  },
  limit: 10
}
```

**Response**:

```javascript
{
  success: true,
  results: [
    {
      id: "1",
      title: "ML Introduction",
      content_preview: "Machine learning is...",
      score: 0.95,
      classification: "Technical Documentation"
    }
  ],
  total: 25,
  processing_time: 0.34
}
```

### POST `/api/generate-insights`

Generate AI insights from the knowledge base.

**Request**:

```javascript
{
  prompt: "What are the main themes in my documents?",
  context_documents: ["1", "2", "3"],
  model: "phi4"
}
```

### MCP Server Endpoints

Each MCP server exposes:

- `GET /health` - Health check
- Server-specific tool endpoints under `/tools/`

## 🔧 Configuration

### Frontend Configuration (`frontend/astro.config.mjs`)

```javascript
export default defineConfig({
  output: 'static',
  adapter: node({
    mode: 'standalone'
  }),
  server: {
    port: 3000
  }
});
```

### Database Connection

Configure in Azure App Service settings or local `.env` file:

```
AZURE_SQL_SERVER=knowledge-sql.database.windows.net
AZURE_SQL_DATABASE=knowledge-base
AZURE_SQL_USERNAME=your_username
AZURE_SQL_PASSWORD=your_password
```

## 🧪 Testing

### Run Health Checks

```bash
```

```
# Test all services
curl https://knowledge-base-orchestrator.azurewebsites.net/health

# Test individual MCP servers
curl https://phi4-mcp-server.azurewebsites.net/health
curl https://knowledge-base-sql-server.azurewebsites.net/health
curl https://knowledge-base-graphrag-server.azurewebsites.net/health
curl https://knowledge-base-search-server.azurewebsites.net/health
```

### Test Document Upload

```bash
curl -X POST https://knowledge-base-orchestrator.azurewebsites.net/api/process-document \
  -F "file=@test-document.pdf" \
  -F "title=Test Document" \
  -F "options={\"enable_classification\":true}"
```

## 📊 Monitoring and Analytics

### Available Metrics

- Document processing times
- Search query performance
- Entity extraction accuracy
- System health status
- User activity patterns

### Database Views

- `vw_document_summary`: Document overview with entity counts
- `vw_entity_relationships`: Knowledge graph relationships
- `vw_popular_searches`: Most frequent search queries

### Stored Procedures

- `sp_GetDocumentWithContext`: Retrieve document with full context
- `sp_SearchDocuments`: Advanced document search with ranking
- `sp_GetKnowledgeGraph`: Extract graph data for visualization

## 🛡️ Security

### Authentication & Authorization

- Azure AD integration for enterprise use
- API key authentication for service-to-service communication
- Role-based access control for sensitive operations

### Data Protection

- Encrypted data storage in Azure SQL Database
- Secure file upload validation
- Content sanitization and validation

### Network Security

- HTTPS-only communication
- Azure App Service security features
- Private endpoints for database access

## 🔮 Future Enhancements

### Planned Features

- **Multi-tenant Support**: Isolated knowledge bases per organization
- **Advanced Analytics**: ML-powered usage analytics and recommendations
- **Real-time Collaboration**: Live document editing and sharing
- **Mobile App**: Native mobile interface for iOS and Android
- **Integration APIs**: Connect with SharePoint, Teams, and other systems

### Scaling Considerations

- **Microservices Architecture**: Further decompose for scale
- **Caching Layer**: Redis for improved performance
- **CDN Integration**: Global content delivery
- **Load Balancing**: Distribute traffic across regions

## 🤝 Contributing

1. Fork the repository

2. Create a feature branch: `git checkout -b feature/new-feature`
3. Commit changes: `git commit -m 'Add new feature'`
4. Push to branch: `git push origin feature/new-feature`
5. Submit a Pull Request

### Development Guidelines

- Follow TypeScript/JavaScript best practices
- Add tests for new functionality
- Update documentation for API changes
- Ensure Azure deployment works correctly

## 📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

## 🆘 Support

### Documentation

- Azure App Service Documentation
- Astro Documentation
- MCP Protocol Specification

### Issues and Questions

- Create GitHub Issues for bugs and feature requests
- Use GitHub Discussions for questions and community support
- Contact the development team for enterprise support

### Troubleshooting

**Common Issues**:

1. **Deployment Failures**
   - Check GitHub Actions logs
   - Verify Azure publish profiles
   - Ensure all secrets are configured
2. **Database Connection Issues**
   - Verify connection strings in App Service configuration

- Check Azure SQL firewall rules
- Test database connectivity

3. **File Upload Failures**
   - Check file size limits (50MB default)
   - Verify supported file formats
   - Monitor App Service logs

---

Built with ❤️ for intelligent knowledge management