# Aviation Knowledge Graph Implementation Guide

## Overview

This package contains a complete implementation of an aviation training knowledge graph document processing system. The system extracts aviation concepts from uploaded documents, identifies relationships between these concepts, and generates a knowledge graph that can be visualized and queried through APIs.

## Problem Being Solved

When users upload aviation training documents, the system should:

1. Extract key aviation training concepts
2. Identify relationships between these concepts
3. Generate a graph data structure (nodes & edges)
4. Store this data in the database
5. Make it available through an API endpoint

## Files Included

The implementation includes the following key components:

1. **Document Analysis Service** (`document-analysis-service.ts`)
   - Processes uploaded documents
   - Extracts text using appropriate libraries (PDF, DOCX, TXT)
   - Triggers concept and relationship extraction
   - Generates and stores knowledge graph data
2. **NLP Service** (`nlp-service.ts`)
   - Contains aviation domain-specific knowledge
   - Extracts concepts using pattern matching and NLP
   - Identifies relationships between concepts
   - Calculates importance scores for concepts
3. **Database Schema** (`database-schema.ts`)
   - Defines tables for documents, nodes, and edges
   - Establishes relationships between tables

- Uses Drizzle ORM for database operations

4. **API Routes**
   - Document Routes (`document-routes.ts`): Handles document uploading and processing
   - Knowledge Graph Routes (`knowledge-graph-routes.ts`): Provides graph data access

5. **Type Definitions**
   - Knowledge Graph Types (`knowledge-graph-types.ts`)
   - Document Types (`document-types.ts`)

6. **Utilities**
   - Logger (`logger.ts`): Consistent logging throughout the application

## Implementation Instructions

### 1. Set Up Dependencies

First, ensure the required dependencies are installed:

bash Copy

```bash
npm install express multer drizzle-orm pg compromise compromise-aviation langchain pdf-parse ma
npm install -D typescript @types/express @types/multer @types/node @types/uuid
```

### 2. Project Structure

Place the files in the appropriate directories:

Copy

```
server/
├── routes/
│   ├── document-routes.ts
│   └── knowledge-graph-routes.ts
├── services/
│   ├── document-analysis-service.ts
│   ├── knowledge-graph-service.ts
│   └── nlp-service.ts
├── database/
│   ├── db.ts
│   └── schema.ts
├── types/
│   ├── document.ts
│   └── knowledge-graph.ts
├── utils/
│   └── logger.ts
└── middleware/
    └── auth-middleware.ts
```

### 3. Database Setup

Create a database connection file (`server/database/db.ts`):

typescript

Copy

```typescript
import { drizzle } from 'drizzle-orm/node-postgres';
import { Pool } from 'pg';

// Create a PostgreSQL connection pool
const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
});

// Create a Drizzle ORM instance
export const db = drizzle(pool);
```

Then run the database migrations using your preferred method (Drizzle CLI or custom script).

### 4. Configure Environment Variables

Set up the following environment variables:

Copy

```
DATABASE_URL=postgres://username:password@localhost:5432/aviation_training
UPLOAD_DIR=uploads
NODE_ENV=production
```

### 5. Application Integration

Update your main application file to use the routes:

typescript Copy

```typescript
import express from 'express';
import documentRoutes from './routes/document-routes';
import knowledgeGraphRoutes from './routes/knowledge-graph-routes';

const app = express();

// Middleware
app.use(express.json());

// Routes
app.use('/api/documents', documentRoutes);
app.use('/api/knowledge-graph', knowledgeGraphRoutes);

// Start server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

### 6. Create Upload Directory

Ensure the uploads directory exists:

bash Copy

```bash
mkdir -p uploads
```

## Usage Examples

### 1. Uploading a Document with Knowledge Graph Processing

typescript Copy

```typescript
// Client-side code (React/TypeScript example)
const uploadDocument = async (file: File) => {
  const formData = new FormData();
  formData.append('file', file);
  formData.append('title', 'Aviation Safety Manual');
  formData.append('description', 'Comprehensive guide to aviation safety procedures');
  formData.append('createKnowledgeGraph', 'true');

  const response = await fetch('/api/documents', {
    method: 'POST',
    body: formData,
  });

  const result = await response.json();
  return result;
};
```

## 2. Retrieving Knowledge Graph Data

typescript Copy

```typescript
// Client-side code (React/TypeScript example)
const getKnowledgeGraph = async () => {
  const response = await fetch('/api/knowledge-graph');
  const result = await response.json();
  return result.data;
};
```

## 3. Visualizing the Graph

You can use libraries like D3.js, Vis.js, or Cytoscape.js to visualize the knowledge graph:

typescript Copy

```typescript
// Example with Cytoscape.js
import cytoscape from 'cytoscape';

const renderGraph = (graphData) => {
  const elements = [
    // Nodes
    ...graphData.nodes.map(node => ({
      data: {
        id: node.id,
        label: node.label,
        category: node.category,
        importance: node.importance
      }
    })),
    // Edges
    ...graphData.edges.map(edge => ({
      data: {
        id: edge.id,
        source: edge.source,
        target: edge.target,
        relationship: edge.relationshipType,
        strength: edge.strength
      }
    }))
  ];

  const cy = cytoscape({
    container: document.getElementById('cy'),
    elements,
    style: [
      // Styling for nodes and edges
    ],
    layout: {
      name: 'cose',
      idealEdgeLength: 100,
      nodeOverlap: 20
    }
  });
};
```

## Troubleshooting

### Common Issues

1. **Document processing status stuck at "processing"**
   - Check server logs for errors
   - Ensure document analysis service is correctly processing the file
   - Verify file permissions on the upload directory
2. **No concepts extracted from document**
   - Check document content quality and format
   - Review NLP service patterns and terms
   - Consider adding more aviation-specific terms to improve detection
3. **Database connection issues**
   - Verify connection string in environment variables
   - Check database user permissions
   - Ensure database schema is correctly migrated
4. **Empty knowledge graph response**
   - Verify document was uploaded with createKnowledgeGraph=true
   - Check document processing status is "complete"
   - Look for errors in document analysis process

## Customization

### Adding More Aviation Concepts

Expand the aviation terms in `nlp-service.ts`:

typescript Copy

```typescript
const AVIATION_TERMS = {
  // Existing categories...

  // Add new category
  airspace_management: [
    'airspace classification', 'special use airspace', 'controlled airspace',
    'uncontrolled airspace', 'restricted area', 'TFR', 'MOA'
  ],
};
```

### Custom Relationship Types

Add new relationship types in `knowledge-graph-types.ts` and `nlp-service.ts`:

typescript Copy

```typescript
// In knowledge-graph-types.ts
export enum RelationshipType {
  // Existing types...
  REQUIRES_CERTIFICATION = 'REQUIRES_CERTIFICATION',
}

// In nlp-service.ts
const RELATIONSHIP_PATTERNS = [
  // Existing patterns...
  {
    type: 'REQUIRES_CERTIFICATION',
    patterns: [
      '% requires certification %',
      '% needs license %',
      'certified for %'
    ]
  },
];
```

## Next Steps & Enhancements

Consider these enhancements for future improvements:

1. **Advanced NLP**: Integrate more sophisticated NLP models like spaCy or Hugging Face Transformers
2. **Real-time Processing**: Add WebSocket support for real-time processing updates
3. **User Feedback**: Allow users to correct or enhance extracted concepts
4. **Visualization Improvements**: Add filtering, grouping, and interactive exploration
5. **Content Recommendation**: Use the knowledge graph to recommend related training materials

## Support

For questions or issues with this implementation, please contact your development team.