

Academic Assignment Helper & Plagiarism Detector (RAG-Powered)

Summary

A candidate will build a comprehensive backend + n8n automation system that:

1. Accepts student assignment uploads via a secured API
2. Processes assignments through n8n for:
 - Text extraction and preprocessing
 - RAG-based research source suggestions
 - AI-powered plagiarism detection against academic database
 - Structured analysis storage in PostgreSQL
3. Uses Docker Compose to orchestrate all services with vector database integration

Security Requirements

- API endpoints must be protected with JWT-based authentication
- Candidate must:
 - Implement `/auth/login` and `/auth/register` (can use hardcoded student credentials)
 - Issue JWT on login with student role permissions
 - Require JWT token to access `/upload` and `/analysis` endpoints

Project Components

1. Backend (FastAPI)

Endpoints

Route	Description	Secured
POST /auth/login	Returns JWT after student credential validation	✗
POST /auth/register	Register new student account	✗
POST /upload	Accepts assignment file, triggers n8n analysis	✓
GET /analysis/{id}	Retrieves analysis results and suggestions	✓
GET /sources	Search academic sources via RAG	✓

On upload:

- Save assignment file locally or pass as binary to n8n
- Call n8n webhook with student metadata and file path
- Return analysis job ID for tracking

2. n8n Workflow

Trigger: Webhook Node

Workflow Steps

1. **Receive Assignment File**
 - Accept file path or binary data from FastAPI
2. **Text Extraction**
 - Use pdf-parse or mammoth (for Word docs) to extract content
3. **RAG Source Search**
 - Query vector database with assignment topic
 - Retrieve relevant academic papers and course materials
4. **AI Analysis (OpenAI/Claude)**
 - **Prompt to extract:**
 - Assignment topic and key themes
 - Research questions identified
 - Academic level assessment

- **Generate:**
 - Relevant source suggestions from RAG results
 - Citation format recommendations
- 5. **Plagiarism Detection**
 - Compare against stored academic papers in vector DB
 - Flag potential plagiarism with similarity scores
- 6. **Structure & Store Results**
 - Transform to structured JSON
 - Insert analysis to PostgreSQL
- 7. **Optional Notifications**
 - Send Slack notification to instructor
 - Email student with analysis summary

3. Database Schema

PostgreSQL Tables

students:

id SERIAL PRIMARY KEY,
 email TEXT UNIQUE,
 password_hash TEXT,
 full_name TEXT,
 student_id TEXT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

assignments:

id SERIAL PRIMARY KEY,
 student_id INTEGER REFERENCES students(id),
 filename TEXT,
 original_text TEXT,
 topic TEXT,
 academic_level TEXT,
 word_count INTEGER,
 uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

analysis_results:

id SERIAL PRIMARY KEY,
 assignment_id INTEGER REFERENCES assignments(id),
 suggested_sources JSONB,
 plagiarism_score FLOAT,

flagged_sections JSONB,
research_suggestions TEXT,
citation_recommendations TEXT,
confidence_score FLOAT,
analyzed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

academic_sources (for RAG):

id SERIAL PRIMARY KEY,
title TEXT,
authors TEXT,
publication_year INTEGER,
abstract TEXT,
full_text TEXT,
source_type TEXT, -- 'paper', 'textbook', 'course_material'
embedding VECTOR(1536) -- for similarity search

4. RAG Implementation

Vector Database Setup

- Use **pgvector** extension in PostgreSQL for embeddings storage
- Pre-populate with academic papers and course materials
- Generate embeddings using OpenAI text-embedding-ada-002

RAG Pipeline

1. **Document Ingestion:** Course materials → text chunks → embeddings → storage
2. **Query Processing:** Assignment text → embedding → similarity search
3. **Context Retrieval:** Top-k relevant sources for AI analysis

5. Dockerized Setup



Services

- **backend:** FastAPI application with JWT auth
- **n8n:** Automation orchestrator with AI and RAG capabilities
- **postgres:** Database with pgvector extension
- **pgadmin:** (Optional) PostgreSQL GUI
- **redis:** (Optional) Caching for embeddings

Environment Variables

OPENAI_API_KEY=your_key_here
POSTGRES_DB=academic_helper
POSTGRES_USER=student
POSTGRES_PASSWORD=secure_password
JWT_SECRET_KEY=your_jwt_secret
N8N_WEBHOOK_URL=http://n8n:5678/webhook/assignment

Project Structure

```
academic-assignment-helper/  
├── backend/  
│   ├── main.py  
│   ├── auth.py  
│   ├── models.py  
│   ├── rag_service.py  
│   ├── requirements.txt  
│   └── Dockerfile  
├── workflows/  
│   └── assignment_analysis_workflow.json  
├── data/  
│   └── sample_academic_sources.json  
├── docker-compose.yml  
├── .env.example  
└── README.md
```

Evaluation Rubric

Skill	What You're Testing
Python Backend	JWT authentication, file handling, RAG implementation, API design
n8n Workflow	Complex automation design, AI integration, error handling, data transformation
LLM/AI Integration	Effective prompting, RAG pipeline, plagiarism detection logic
Database Design	Schema design, vector database usage, relationship modeling
RAG Implementation	Vector embeddings, similarity search, context retrieval, knowledge base setup

**Docker
Orchestration**

Multi-service setup, networking, environment management,
pgvector setup

Deliverables

1. **GitHub Repository**

Include complete project with:

- Docker setup (docker-compose.yml with pgvector)
- Backend source code with RAG implementation
- n8n workflow export (.json)
- Sample academic data for testing
- .env.example with all required variables
- Comprehensive README with setup instructions

2. **5-Minute Demo Video**

Demonstrate:

- Student registration and JWT authentication
- Assignment upload process
- n8n workflow execution walkthrough
- RAG-based source suggestions in action
- Plagiarism detection results
- Database storage verification
- Docker services interaction

3. **Technical Documentation**

- RAG pipeline architecture explanation
- AI prompting strategy documentation
- Database schema reasoning
- Security implementation details

Submit your work on this email:-xender18lab@gmail.com