🚀 AI ENGINEERING ASSIGNMENT: HANDBOOK GENERATOR

📬 Submission Guide

Submit your completed work to:

📧 **tk.lunartech@gmail.com**

What to Include

Item	Description
Code	GitHub repository link or `.zip` archive
Setup Guide	Clear instructions to run your application
Demo	Short video walkthrough OR screenshots showing the app working
Write-up	Brief summary: what you built, approach taken, any challenges

Submission Checklist

- ☐ Working application (local or deployed)
- ☐ Can upload PDF documents
- ☐ Can chat and receive contextual responses
- ☐ Can generate a 20,000-word handbook via chat
- ☐ Demo video or screenshots included

Executive Summary

What is This Assignment?

You are tasked with building a **simple AI-powered chat application** that can generate **20,000-word handbooks** from uploaded PDF documents. This is an AI engineering exercise designed to evaluate your ability to integrate modern AI tools and techniques into a functional application.

Why This is Simpler Than It Looks

This assignment is **not** about building everything from scratch. We are providing you with:

1. **A complete research paper** explaining the LongWriter technique—the method used to generate extremely long-form content that exceeds typical LLM output limits
2. **A full reference implementation** with working code that demonstrates the core generation logic
3. **Clear architectural guidance** on exactly which tools to use and how they connect
4. **Permission to use AI coding assistants**—this is an AI engineering task, so using tools like KiloCode, Antigravity, Cursor, or Kimi K2.5 is not only allowed but encouraged

You are essentially **assembling proven components** into a working application. The hard research and implementation work has already been done.

The Core Task

Build an application where a user can:

1. **Upload PDF documents** (research papers, documentation, any text-based PDFs)
2. **Chat with the system** to ask questions about the uploaded content
3. **Request a handbook** and receive a 20,000+ word structured document generated from the PDF content

The entire interaction should happen through a simple chat interface. No complex UI required.

Technology Stack Overview

Component	Technology	Purpose
Frontend	Gradio, Streamlit, or any UI	Simple chat interface
LLM	Grok 4.1	Long-context generation with LongWriter technique

Component	Technology	Purpose
RAG System	LightRAG	Knowledge graph creation from PDFs
Database	Supabase	Vector storage for embeddings
PDF Processing	PyPDF, pdfplumber	Extract text from uploads

What Success Looks Like

A successful submission demonstrates:

- **PDF → Knowledge**: Upload a PDF and have its content indexed
- **Chat → Context**: Ask a question and get a response that references the PDF content
- **Request → Handbook**: Ask for a handbook and receive a 20,000+ word document with structure, headings, and content derived from the uploaded materials

---

🤖 AI Tools: Why You Should Use Them

> **We don't just allow AI coding tools—we expect you to use them.**

Why We Encourage AI Assistance

This is an **AI engineering assignment**. The ability to effectively leverage AI tools is a core competency we're evaluating. In the real world, AI engineers use AI to:

- **Accelerate development** — Write boilerplate, debug faster, explore APIs
- **Learn new technologies** — Get up to speed on LightRAG, Supabase, Grok quickly
- **Focus on architecture** — Let AI handle syntax while you design the system
- **Iterate rapidly** — Prototype ideas without getting stuck on implementation details

If you complete this task entirely by hand, you're working harder, not smarter. That's not the mindset we're looking for.

Recommended Tools

Tool	Best For	Why Use It
KiloCode	VS Code users	Inline completions, context-aware suggestions

Tool	Best For	Why Use It
Antigravity	Complex multi-file tasks	Agentic coding, understands full project context
Cursor	End-to-end development	AI-first editor with powerful chat + apply
Kimi K2.5 Thinking	Tricky logic problems	Deep reasoning for algorithmic challenges
Claude / GPT / Grok	General Q&A	Debugging, explaining concepts, code review

How to Use Them Effectively

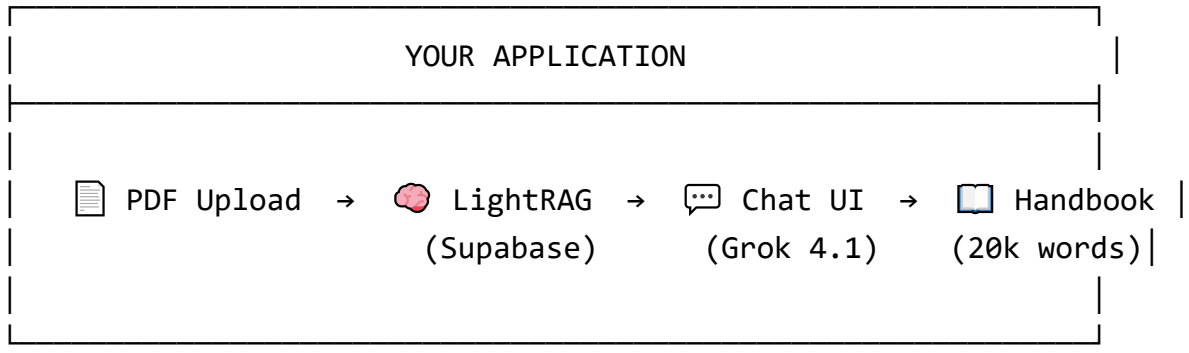
1. **Give context** — Share error messages, explain what you're trying to build
2. **Iterate** — Don't expect perfect code on first try; refine with follow-ups
3. **Verify** — AI can make mistakes; test everything it generates
4. **Learn** — Understand what the AI produces, don't just copy-paste blindly

Bottom line: Use every tool at your disposal. That's what a real AI engineer does.

🎯 The Task (One Sentence)

Build a chat app where you upload PDFs, ask questions, and generate a 20,000-word handbook—all through conversation.

What You're Building



Core Flow

1. **User uploads PDFs** → System extracts and chunks text
2. **Chunks stored in Supabase** → LightRAG creates knowledge graph
3. **User chats with UI** → Context retrieved from knowledge graph
4. **LongWriter generates** → 20,000-word handbook output

📦 What You're Given

1. Research Paper

`Documentation/Unleashing 10000 Word Generation From Long Context.pdf`

Read this first. It explains the LongWriter technique for generating ultra-long content.

2. Reference Implementation

`LongWriter-main/` — Complete codebase including:

- `agentwrite/` — Core generation logic
- `train/` — Training scripts (for reference)
- `trans_web_demo.py` — Web UI demo

3. Required Tools

Tool	Purpose
Grok 4.1	Long-context LLM for generation
LightRAG	Knowledge graph from PDFs
Supabase	Vector storage + database
Any AI IDE	KiloCode, Antigravity, Kimi K2.5, etc.

🛠️ Technical Requirements

Stack

```
Frontend:       Any (React, Vue, Streamlit, Gradio)
Backend:        Python or Node.js
LLM:            Grok 4.1 (via API)
RAG:            LightRAG
Storage:        Supabase (pgvector)
```

Must-Have Features

1. **PDF Upload** — Accept and parse PDF files
2. **Knowledge Graph** — Store content in LightRAG
3. **Chat Interface** — Simple text input/output
4. **Handbook Generation** — Generate 20,000+ words via chat

Optional Features

- Multiple PDF support
- Progress indicator for long generation
- Export to Markdown/PDF
- Conversation history

🧪 Test Case

Input:

- Upload 2-3 AI-related PDFs
- Chat: “Create a handbook on Retrieval-Augmented Generation”

Expected Output:

- 20,000+ word structured document
- Table of contents
- Sections with proper headings
- Citations from uploaded PDFs

🔒 Confidentiality Notice

This assignment, including all provided materials, is the intellectual property of **LunarTech**. By participating:

- You agree not to share materials publicly
- You may not publish the solution without permission
- This is for evaluation purposes only

Good luck! This is simpler than it looks. The research is done, the code exists—you're putting pieces together.