# Proposal: Backend Developer for LLM-Powered Multi-Tenant AI Assistant

## Project Title
Backend Developer for LLM and API Integration Project

## Project Understanding
Client wants to build a multi-tenant AI assistant feature within their SaaS platform, allowing each of their own clients to:
- Upload their own knowledge base (PDFs, docs, etc.)
- Get a custom AI chatbot trained on this data (ring-fenced per tenant)
- Embed the chatbot directly inside LinkedIn messaging
- Enable multi-turn conversations, persona config, retry logic, and fallback responses
- Perform data retraining on new uploads
- Set bot personality settings per tenant (e.g., tone, temperature, style)

## Key Queries Before Finalization
1. What document types will be uploaded? PDF, DOCX, text, URLs?
2. Should bots initiate LinkedIn conversations or only reply?
3. Do bots need to maintain long-term memory or session-only?
4. Will each tenant configure personality (e.g., tone, GPT temperature)?
5. Expected number of clients in first 6 months?
6. Will uploaded documents be stored on S3/MinIO or internal storage?

## Tech Stack (Suggestion)
Backend Framework: Django
LLM API: OpenAI (GPT-4 / GPT-3.5)
Vector DB: Qdrant
Task Queue: Celery + Redis
Retrieval Technique: RAG (Retrieval-Augmented Generation)
Integration Layer: LinkedIn API + OAuth tokens per tenant
Multitenancy: Isolated DB per tenant (via router)

## Key Features

- Isolated DB for Multitenancy: Full DB per tenant using Django DB router
- Document Ingestion Pipeline: Parsing, chunking, embeddings
- Multi-turn LLM Chatbot: RAG + memory + retry handling
- Bot Persona Config: Per-tenant system prompts (tone, behavior, limits)
- LinkedIn Integration: OAuth token, send/receive with rate-limit fallback
- Custom Retraining: On-demand re-embedding after new uploads

## Development Timeline

| Phase | Adjusted Duration |
|---|---|
| Project Setup | 3 days |
| File & Data Ingestion | 8 days |
| RAG-based Bot Engine | 10 days |
| Bot Persona + Config Panel | 5 days |
| LinkedIn API Integration | 9 days |
| Custom Retraining Logic | 4 days |
| Admin & Client Panel APIs | 4 days |

**Total Estimated Duration: ~43 Working Days (~9 weeks)**

## Development Process (As per suggested Tech Stack)

This section outlines the high-level process flow and steps to be followed during the development of the project:

1. Environment Setup: Set up Django project with PostgreSQL, Celery and Redis

2. Multi-Tenant Setup: Configure dynamic database router for isolated DB per tenant.

3. Authentication & Tenant Access: Implement JWT-based login and tenant-level permission middleware.

4. File Upload & Storage: Build APIs for document upload and store documents to disk or S3.

5. Document Processing: Extract text from PDFs/DOCX, clean it, chunk it, and prepare for embeddings.

6. Vector Embedding & Storage: Generate embeddings using OpenAI and store in Qdrant with tenant key.

7. Bot Engine Integration: Implement Retrieval-Augmented Generation using Qdrant + OpenAI completion API.

8. Bot Persona Configuration: Allow each tenant to set bot behavior (prompt template, temperature, tone).

9. LinkedIn API Setup: Enable LinkedIn OAuth per tenant and securely store access tokens.

10. LinkedIn Message Handling: Use Celery tasks to listen for and respond to LinkedIn messages.

11. Custom Retraining Logic: Trigger re-embedding pipeline on document upload/update automatically.

12. Admin and Client Dashboards: Expose APIs to fetch usage logs, configuration, and bot history.