ТЕХНИЧЕСКОЕ ЗАДАНИЕ (ТЗ)

На разработку прототипа локальной системы генеративного поиска (RAG) по корпоративной базе знаний

1. Общие сведения и цель проекта

Наименование системы: Корпоративный ИИ-Ассистент (On-Premise Knowledge Bot).

Назначение: Автоматизация поиска ответов, суммаризации и анализа внутренних текстовых документов компании в защищенном контуре.

Цель создания прототипа: Проверить применимость Open-Source языковых моделей (LLM) для ответов на специфичные вопросы по внутренним регламентам без передачи данных в публичные облачные сервисы.

2. Архитектура и технические требования

2.1. Требования к развертыванию и ИБ (Критически важно)

Локальный контур (On-Premise): Все компоненты системы (векторная база данных, LLM-модель, парсеры и бэкенд) должны запускаться локально на выделенном сервере/рабочей станции (или в приватном облаке компании).

Запрет внешних API: Запрещено использование API OpenAI, Anthropic, Cohere и других проприетарных облачных моделей.

Конфиденциальность: Система не должна отправлять телеметрию или логи работы внешним вендорам.

2.2. Требования к компонентам ИИ (Архитектура RAG)

Языковая модель: Использование квантованных Open-Source моделей (например, Llama-3-8B-Instruct, Mistral-7B-Instruct, Qwen-2.5-7B или аналогичных), способных эффективно работать на доступных ресурсах (например, 1-2 потребительские GPU уровня RTX 3090/4090 или серверные T4/A10).

Эмбеддинги: Использование легковесных моделей, поддерживающих мультиязычность (казахский/русский), например, семейства multilingual-e5.

Векторное хранилище: Использование бесплатных open-source решений для индексации документов: ChromaDB, FAISS, Qdrant или Milvus.

3. Функциональные требования

Система в рамках прототипа должна реализовывать следующий минимальный функционал (MVP):

3.1. Модуль загрузки и обработки документов (Ingestion Pipeline)

Поддержка форматов: .pdf, .docx, .txt.

Автоматическое разбиение документов на логические блоки (Chunking) с перекрытием (Overlap) для сохранения контекста.

Генерация векторных эмбеддингов для каждого блока и сохранение их в векторную БД.

3.2. Чат-интерфейс пользователя (UI)

Текстовое поле для ввода вопроса на русском или казахском языках.

Окно отображения диалога (в стиле классических LLM-чатов).

Обязательное требование: Вывод ссылок/цитат на первоисточники (конкретный документ и номер страницы/абзаца), на основе которых модель сформулировала ответ.

Кнопка «Сбросить контекст» для очистки истории текущего диалога.

3.3. Админ-панель (упрощенная)

Простой интерфейс (или скрипт) для загрузки новых документов в базу знаний и удаления старых.

4. Требования к качеству работы (Метрики)

Релевантность ответов: Модель должна отвечать строго на основе загруженных документов. Если в документах нет ответа на вопрос, модель должна честно отвечать: *«В предоставленных документах нет информации по данному вопросу»*, минимизируя галлюцинации.

Языковой барьер: Модель не должна смешивать языки (отвечать на русском, если спросили на казахском, и наоборот).

Скорость ответа (Latency): Время генерации первого токена (Time to First Token) не должно превышать 3-5 секунд на тестовом железе.

5. Этапы выполнения и формат сдачи работ

Этап 1: Подготовка окружения. Разработчик разворачивает базовую архитектуру на своих мощностях и демонстрирует готовность инфраструктуры.

Этап 2: Тестирование на демо-данных. Компания предоставляет исполнителю тестовый (открытый) пакет документов (до 10-15 регламентов/инструкций).