PIOTR SYNAK, PhD

AI & DATA SYSTEMS ARCHITECT | R&D LEADER

piotr.d.synak@gmail.com   Winterthur, Switzerland

A LIST OF RECENT AI PROJECTS:

1. Decreen. AI-powered knowledge graph and onboarding assistant

Decreen is a multi-service platform designed to help teams understand and navigate complex software landscapes. It consolidates fragmented knowledge from Jira, Confluence, and GitHub into a single, queryable knowledge graph, enabling reliable onboarding, system exploration, and architectural reasoning. In addition to semantic search and AI-assisted querying, Decreen automatically generates and maintains living architecture diagrams, allowing engineers, architects, and product leaders to understand systems, APIs, teams, and flows without relying on manually maintained documentation.

The platform synchronizes content from external systems, processes it through AI-assisted extraction and enrichment pipelines, and assembles structured entities and relationships into a graph-backed data model. Insights are exposed via REST and GraphQL APIs as well as a modern React-based user interface with dashboards, project exploration views, and graph visualizations. LLMs are used in constrained, pipeline-driven roles such as information extraction, classification, summarisation, and generation of personalised onboarding tasks from live organisational context.

Tech stack:

- Frontend: React 18, Vite, Material-UI, data/graph visualisation libraries (e.g., Nivo, Recharts, React Flow, Three.js)
- Backend REST API: Node.js, Express, Sequelize
- GraphQL and services: Python, FastAPI, Strawberry GraphQL
- Background processing: Celery workers and pipelines for extraction and enrichment
- Database: PostgreSQL with pgvector (vector similarity search)
- Caching/queues: Redis
- LLMs and embeddings: OpenAI GPT models and optional Hugging Face; embeddings + RAG over the knowledge graph
- Auth and integrations: Auth0 SSO; Atlassian (Jira, Confluence) and GitHub connectors
- Dev/infra: Docker/Compose, comprehensive test suites (Jest/Vitest/Pytest)

LLM models:

- OpenAI GPT models for reasoning tasks, extraction, summarisation, and classification, e.g. GPT-4.1 or GPT-4o
- OpenAI text embedding models for semantic search, e.g. text-embedding-3-large or text-embedding-3-small
- Optional Hugging Face models via on-device transformers for cost-sensitive or offline scenarios, e.g. sentence-transformers all-MiniLM-L6-v2 accessed through a lightweight runtime

2. On-prem RAG system

Local RAG System is a production-ready, containerised platform for Retrieval-Augmented Generation. It lets users upload documents, builds vector indexes for semantic search, and answers queries via an AI assistant with switchable backends (baseline RAG, LangGraph agent, or LlamaIndex agent). The system includes a React web UI, a FastAPI backend, a dedicated LLM inference microservice, and supporting data services behind an Nginx reverse proxy. It ships with Docker Compose stacks for development, testing, and production, with health checks, logging, and optional SSL.

Tech stack:

- Frontend: React 18, Vite, Material UI
- Backend: FastAPI (Python), structured logging, global exception handling
- LLM service: Python FastAPI app with model adapters (Hugging Face ecosystem, Sentence Transformers), optional CUDA acceleration
- Agent options: LangGraph, LlamaIndex, and a baseline RAG engine (runtime-switchable)
- Data layer: PostgreSQL (metadata), ChromaDB (vector store), Redis (cache/session)
- Infrastructure: Nginx reverse proxy, Docker and Docker Compose (profiles for dev/prod/test), optional tools like Jupyter, Prometheus/SonarQube in certain profiles
- Security and operations: JWT/CORS settings, environment-based config, health endpoints, and optional TLS/SSL

LLM models:
- Llama 2 7B Chat (meta-llama/Llama-2-7b-chat-hf) — GPU-optimized text-generation/chat
- Mistral 7B Instruct (mistralai/Mistral-7B-Instruct-v0.1) — GPU-optimised instruction following; uses 8-bit loading by default for efficiency
- GPT-2 Large (gpt2-large) — lightweight CPU-friendly fallback
- Embeddings: all-MiniLM-L6-v2 (Sentence Transformers) for vector search
- Model management: a registry-driven, adapter-based LLM service selects the active model at runtime and supports background loading, health/readiness checks, and per-model configuration

3. Hedge Fund AI Analyst

The project delivers an AI Analyst platform that automates a 28-step hedge fund investment research workflow end-to-end. It ingests financial documents, orchestrates multi-provider LLM analysis, tracks progress in real time, and outputs briefing papers and workflow results with enterprise security and multi-tenant isolation.

Tech stack:
- Frontend: React 18 with TypeScript, Vite, Material UI, React Router, Redux Toolkit, and real-time updates via WebSockets (Socket.IO client).
- Backend: FastAPI (Python 3.12), Pydantic v2, SQLAlchemy 2, Alembic, JWT authentication, Uvicorn/Gunicorn.
- Data and infrastructure: PostgreSQL 16 with Row-Level Security, Redis caching, Nginx reverse proxy, Docker and Docker Compose for deployment.
- Document processing: PyPDF2, Unstructured, pdf2image, Tesseract OCR.
- Observability and testing: Prometheus client metrics, structured logging (Loguru/structlog), pytest and Vitest.

LLM models:
- OpenAI: GPT-4, GPT-4 Turbo, GPT-3.5 Turbo.
- Anthropic: Claude-4 Opus, Claude-4 Sonnet, Claude-4 Haiku.
- Google: Gemini Pro, Gemini Pro Vision.
- Orchestration: Multi-LLM routing with A/B testing across models; the demo defaults to Anthropic Claude for initial press-release analysis.

4. KSA Hospital Bed Occupancy Forecasting System

The project delivers a per-ward forecasting system that automates a 7-stage hospital bed-occupancy workflow end-to-end for Kantonsspital Aarau. It ingests raw hospital exports and regional calendars, builds per-ward daily time series, trains hybrid Prophet+LSTM models, and orchestrates forecasting, validation, and auto-calibration in a single pipeline. The system produces 7-day ward- and hospital-level capacity forecasts with alert thresholds, supports fast daily production runs and scheduled execution on a single hospital server, and is tuned for operational decision-making by clinical and management teams.

Tech stack:
- **Data pipeline and orchestration:**
  Python 3.8 CLI pipeline with modular scripts for data transformation, per-ward aggregation, model training, forecasting, validation, and auto-calibration; central orchestration via command-line flags for full runs, daily production runs, retraining-only runs, and custom forecast dates.
- **Modeling and forecasting:**
  Hybrid time-series stack combining Prophet for daily admissions, TensorFlow/Keras LSTM for daily discharges, conservation-based MC dynamics, and ward-specific configuration with a 7-day forecast horizon.

- **Validation and calibration:**
  Weekly validation of forecast quality with MAE, RMSE, MAPE and bias metrics per ward and hospital total; per-ward auto-calibration engine that compares recent vs historical windows, applies dynamic variance- and percentile-based guardrails, incorporates seasonal and holiday patterns, and re-generates forecasts without retraining when only post-processing parameters need adjustment.