Project Portfolio (2018–Present) — Salmon Joy | AI Engineer, DTBX Innovate

Intro

This document is a consolidated summary of the key projects I have delivered from 2018 to the present, across SmartByte Computer Education, GITAM, and DTBX Innovate. It highlights the intent, outcomes, and execution approach for each project in a structured format. The portfolio covers work spanning AI/ML, LLM systems, RAG chatbots, data pipelines, model tuning, deployment, automation, and governance tooling.
I am Salmon Joy, currently working as an AI Engineer at DTBX Innovate, and this list represents the most relevant work I have completed till date.

Banking, LLM, RAG, Compliance (Core)

1) Action Points to Bank Process Notes (DSPy + CoT)
What I was trying to achieve: Convert regulator circular text into structured action points and bank-ready process notes with consistent reasoning.
What I achieved: Reduced manual drafting time by **65%** and improved reviewer acceptance to **85%** per draft iteration cycle.
How I achieved it: Used **CoT-style structured prompts**, **DSPy prompt tuning**, and schema-validated outputs to keep notes consistent and auditable.

2) High-Quality Training Data Pipeline (Gemini Generator + Judge)
What I was trying to achieve: Generate high-quality LLM/embedding training data with strong positives and hard negatives for banking compliance.
What I achieved: Generated **75,000+** samples initially and scaled dataset to **150,000** samples, including **68,000** hard negatives for robust training.
How I achieved it: Built a generator-judge pipeline with **Gemini Pro/Flash**, rejection sampling, hard-negative mining via retrieval, and automated stats + QA checks.

3) Domain Embeddings Improvement (LoRA on Nomic Embedder)
What I was trying to achieve: Improve retrieval relevance for banking queries using a domain-tuned embedding model.
What I achieved: Improved top-k retrieval relevance by **12%** (offline eval) and reduced irrelevant chunk picks by **20%** for compliance queries.
How I achieved it: Trained LoRA adapters on **banking Q/A pairs + hard negatives**, tracked retrieval metrics, and deployed the tuned embedder in the RAG stack.

4) Guardrails for LLM Powered Postgres Query (Native Language + Translation)
What I was trying to achieve: Let users query Postgres in natural language (including regional languages) with safe and correct SQL generation.
What I achieved: Reduced unsafe/invalid SQL attempts by **90%** and increased successful query execution to **96%** in controlled testing.
How I achieved it: Implemented SQL allowlist rules, schema-aware generation, query validation,

parameterization, and **translation layer** for regional languages.
Link: Prework Github Repo

5) End-to-End RAG Chatbot (Dense + Sparse BM25, Django + Qdrant, Docker)
What I was trying to achieve: Build and deploy a scalable chatbot with accurate retrieval using dense embeddings + sparse BM25 for banking content.
What I achieved: Improved answer grounding rate to **88%** and reduced "no-result" queries by **35%**, with **p95 latency ~2.4s** on deployment.
How I achieved it: Implemented hybrid retrieval (dense + BM25), Django streaming endpoints, Qdrant collections, caching, and Docker-based deployment.
Link: Chatbot Github Repo

6) API Scoring System from Log Data (Django)
What I was trying to achieve: Score APIs on governance and quality dimensions using observed traffic/log evidence.
What I achieved: Automated scoring across **120+ APIs** daily and detected **PII leakage patterns** in **8%** of sampled payloads during initial rollout.
How I achieved it: Built rule engines for sensitive data detection, naming/versioning checks, auth/access patterns, SQLi signals, stability scoring, and dashboards in Django.
Link: Prework Github Repo (Not the actual project repo just experimentation)

7) Data and Model Quality Tools
What I was trying to achieve: Improve dataset reliability and model comparisons with repeatable QA checks.
What I achieved: Removed **14% duplicate/near-duplicate** samples and improved training stability (variance reduction) by **25%** across runs.
How I achieved it: Added dedup pipelines, distribution drift checks, label balance reports, experiment tracking, and model comparison harnesses.

8) Circular Actionables → Department Classification Backend (SVM + MLP on Embeddings)
What I was trying to achieve: Automatically classify extracted circular action points into the **most relevant bank department(s)** so routing and ownership becomes faster and consistent.
What I achieved: Classified **40,692 actionables across 18 departments** with **~92% overall accuracy** and improved routing productivity by **~70%** by removing manual tagging effort.
How I achieved it: Generated embeddings for each actionable, trained **SVM and MLP** classifiers, performed **hyperparameter tuning with Optuna**, compared models using validation metrics, and deployed the best model behind a backend API for batch + real-time classification.
Link: Prework for model training (Not the actual project repo just experimentation)

LLM Infra and Deployment

9) LLM Deployment via Ollama (GPU + Wrappers + Monitoring)
What I was trying to achieve: Standardize LLM deployment on Azure Cloud with consistent request/response formats and system monitoring.
What I achieved: Cut integration time by **50%** and maintained stable serving at 1**6–25 req/min** with GPU, plus

CPU/RAM monitoring alerts.
How I achieved it: Deployed models on Ollama, wrote wrapper APIs for unified schema, implemented health checks, logging, and resource monitoring.

Research and Advanced Systems (Healthcare Digital Twin / FedPINN)

Link: Documentaion Github Repo

10) Feed-Forward Network for Weight Computation (Kubernetes, Multi-modal)
What I was trying to achieve: Compute adaptive weights for multi-modal inputs efficiently in a scalable containerized environment.
What I achieved: Improved throughput by **1.7x** and reduced compute cost by **30%** using optimized batching and scaling.
How I achieved it: Designed FFN weighting module, deployed on Kubernetes with autoscaling, and optimized pipelines for multi-modal feature flows.

11) Adaptive FedPINN for Distributed Prediction (Endometriosis)
What I was trying to achieve: Build federated PINN architecture for distributed clinical data prediction with stable convergence.
What I achieved: Achieved **18% lower prediction error** and **1.5x faster convergence** compared to baseline training setups.
How I achieved it: Designed adaptive aggregation, handled non-IID data behavior, and tuned physics-informed loss balancing for convergence.

12) Digital Twin Framework for Endometriosis (3D + Explainability + Clinical Validation)
What I was trying to achieve: Deliver an interpretable, clinically usable prediction system with a uterus digital twin simulation.
What I achieved: Produced a working framework delivering **explainable risk indicators** and a 3D simulation pipeline with **clinical validation-ready outputs**.
How I achieved it: Combined optimized FedPINN training, explainability methods, and a digital-twin simulation layer aligned to clinician interpretation needs.

Products and Automation

13) Smarty eLearning App (Flutter + Content + LLM Doubt Solving, Hybrid RAG)
What I was trying to achieve: Build an app for learning (video/pdf/quiz) plus a chat system that answers doubts in a student-persona style.
What I achieved: Improved student doubt resolution rate to **82%** within chat and reduced support burden by **55%** with RAG-based answers.
How I achieved it: Built Flutter app and React + NodeJS admin panel, integrated RAG with Qdrant hybrid retrieval, and created persona-driven prompting for explanations.
Link: App Playstore

14) n8n Automation on VPS (Social, Email Routing, Enquiries, Telegram Ops)
What I was trying to achieve: Automate social media handling, enquiry responses, email segmentation, and internal task routing.
What I achieved: Automated **25+ workflows**, reduced manual follow-ups by **60%**, and improved response SLA to **under 15 minutes** for common enquiries.
How I achieved it: Hosted n8n on VPS, built triggers/actions across LinkedIn/Instagram/email/Telegram, and added routing rules per department.
Link: Self hosted N8N

15) AI Toolchain for Productivity (Codex, Heygen, ElevenLabs, NapkinAI)
What I was trying to achieve: Speed up development and content creation using modern AI tools with a repeatable workflow.
What I achieved: Reduced coding iteration time by **35%** and cut video creation time by **50%** for training/content deliverables.
How I achieved it: Standardized scripts/templates, integrated tool outputs into one pipeline (text → voice → video → assets), and added QA checkpoints.
Link: Education videos made using tools

ML / CV / Full-Stack Projects

16) Face Recognition System
What I was trying to achieve: Build reliable identity verification using face embeddings.
What I achieved: Delivered **96.4% verification accuracy** with **~55 ms/face** matching and **~70% reduction** in manual checks.
How I achieved it: OpenCV + face encodings, similarity thresholds, and iterative tuning with test split validation.
Link: Prework and experimentation Repo

17) Custom YOLOv8 Classification (Full-Stack)
What I was trying to achieve: Train and deploy a usable classification pipeline with dataset management + inference UI.
What I achieved: Achieved **Top-1 93.2%** with **mAP@0.5 0.89**, supporting **~22 images/sec** inference on GPU.
How I achieved it: Fine-tuned YOLOv8, built React + Node/Express + Postgres, and shipped inference APIs + dashboards.

18) Injury Prediction in Gymnastics
What I was trying to achieve: Predict injury risk/type for early preventive actions.
What I achieved: Achieved **F1 0.82** with **Recall 0.86**, improving high-risk detection by **35%** over baseline rules.
How I achieved it: Feature engineering, Random Forest/SVM training, tuning, and stratified evaluation.

19) Student Feedback Classification (BERT)
What I was trying to achieve: Auto-tag feedback by sentiment/category and surface insights to admins.
What I achieved: Achieved **Accuracy 94.1%** and **Macro-F1 0.91**, reducing review time by **60%**.

How I achieved it: BERT-based classification, full-stack app with Postgres, and dashboard-driven feedback workflows.

20) Student Marks Prediction (SVM)
What I was trying to achieve: Predict marks and identify at-risk students early.
What I achieved: Achieved **MAE 4.2** and **R^2 0.78**, enabling earlier intervention by **2–3 weeks**.
How I achieved it: SVM regression pipeline, standardized features, and analytics dashboards.

21) Skill Matrix Website (GITAM)
What I was trying to achieve: Capture skills, compute gaps, and generate improvement insights.
What I achieved: Supported **160 users** and 1**65 skill categories**, cutting reporting time by **80%**.
How I achieved it: React + Node/Express + Postgres with role-based dashboards and gap analysis logic.
Link: PoC project Repo