PROJECT PORTFOLIO · JUNE 2026

Marc Hunter Eppley

Biotech professional working at the intersection of wet-lab science, applied AI, and data engineering. Selected work below.

Winston-Salem, NC · hunter.eppley.pro@gmail.com · 336-529-7404 · github.com/Sleepytimebaby

---

Six projects spanning *regenerative-medicine ecosystem analytics, multi-LLM measurement pipelines, bioinformatics, EEG signal analysis, deep-learning sentiment models, and edge computer vision*. Each one solves a real problem end-to-end.

CONTENTS

01 Situational Search — AI Investor Matcher PTRME / WFIRM

`Python · Streamlit · OpenAI API · pandas · Plotly`

Problem

PTRME's regenerative-medicine ecosystem includes hundreds of biotech companies, each at different funding stages — but manual triage of which investors to introduce to which company was slow and inconsistent.

What I built

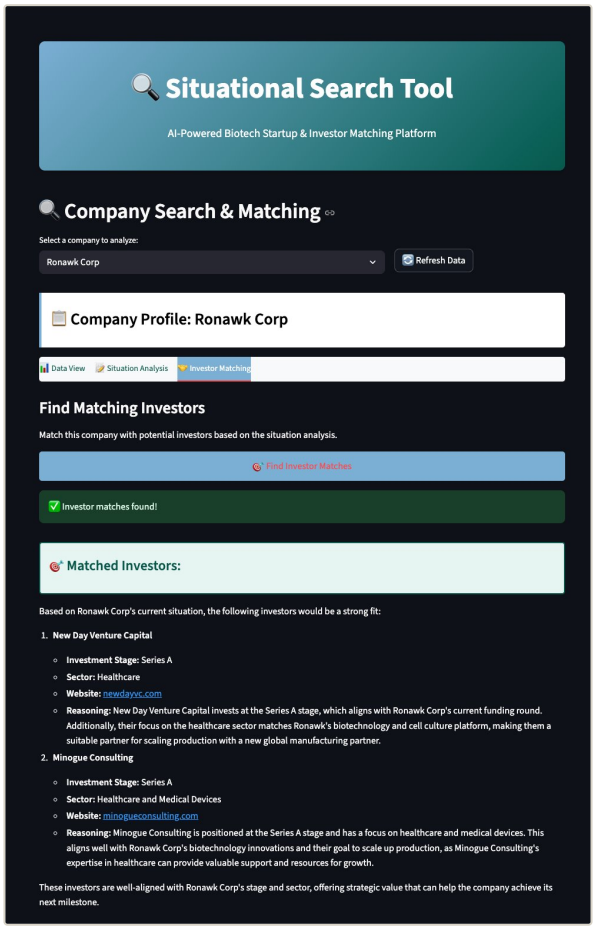
A **Streamlit web application** that ingests PTRME's company-profile and investor datasets, uses the OpenAI API to generate a three-sentence situational analysis per company — identifying sector and funding stage from pre-seed through Series C — and then surfaces investor matches against that analysis. Custom-styled UI with PTRME branding, interactive dashboards, and persistent session state across the analysis workflow.

Outcome

Live tool for PTRME's ecosystem team. Separately, **automated Creatio CRM data entry for 900+ company profiles**, eliminating weeks of manual ingest work and standardizing the underlying dataset.

Source

`Code held under WFIRM repository — available on request`



Investor matching for "Ronawk Corp" — AI-generated investor matches with stage, sector, and reasoning

02 Generative Engine Optimization (GEO) Pipeline CONFIDENTIAL CDMO

`Python · OpenRouter · OpenAI / Anthropic / Google / Perplexity · openpyxl`

Problem A CDMO client needed to measure whether their brand surfaces in AI-model responses to real buyer-style prompts, and which competitors appear in their place — a new flavor of SEO for the LLM era.

What I built A production Python pipeline that queries **four frontier LLMs** (OpenAI, Anthropic, Google, Perplexity) through the OpenRouter API and writes structured, analyst-ready results to an Excel workbook. Designed a **57-prompt research library** across query clusters (Discovery, Comparison, Selection, Thought Leadership), enforced 20 priority-keyword coverage, and built customer size and stage variants to stress-test brand visibility against 16 named competitors.

Outcome Production-grade reliability: response validation rejects silent truncations, automatic retries handle transient model errors, and every run produces a timestamped backup before any writes. Delivers analyst-ready data for monthly brand-visibility tracking.

Source `Private — code and prompt library available on request under NDA`

03

DNA Sequence Analyzer with Protein Translation

`Python · Biopython · FASTA · CSV`

Problem — Bench biologists frequently need a comprehensive single-pass summary of a FASTA-formatted sequence — nucleotide counts, GC content, restriction sites, and the encoded protein — without stitching together five separate tools.

What I built — A Python bioinformatics toolkit that parses multi-sequence FASTA files via Biopython's SeqIO and computes, for each sequence: **nucleotide counts (A/T/C/G), GC content, reverse complement, start-codon positions, restriction-enzyme cut sites (EcoRI, BamHI, HindIII, NotI, SpeI), and standard-genetic-code protein translation** (terminating at the first stop codon). Outputs a flat CSV ready for downstream analysis.

Outcome — Validated on the SARS-CoV-2 reference genome (FASTA). One run produces a complete bioinformatics summary per sequence — drop-in ready for cell-line work, restriction-cloning planning, or expression-construct design.

Source — `github.com/Sleepytimebaby/DNA-Sequence-Analyzer-with-Protein-Translation`

Sequence	*Nucleotides (A / T / C / G)*	*GC%*	*Start codons*	*Restriction cut sites*	*Protein translation*
CDS-1	A: 2938, T: 4245, C: 2008, G: 2734	39.77%	338	EcoRI: [4472], BamHI: [753], HindIII: [1248, 3171, 7614, 8244, 10240, 10966], SpeI: [4452, 5091]	MANQYVLRVADCTNVYYTRLWSSREAVSVYGAAAACGF… *(3974 aa)*
CDS-2	A: 2129, T: 2736, C: 1410, G: 1741	39.31%	236	EcoRI: [2352, 5450], HindIII: [2589, 5463, 6252, 7581]	EPCSEHHVIRAFDIYNKDVACITKFPKINCVRFRNTGM… *(2671 aa)*
CDS-3	A: 894, T: 1147, C: 632, G: 708	39.63%	96	SpeI: [2385, 2964]	MALIFVLMLITLYRCPFVLCNFQVCTDQLRQQEVYLPN… *(1126 aa)*
CDS-4	A: 147, T: 235, C: 130, G: 133	40.78%	10	none detected	MIGGLFSVGFEQFIQHANVTTGGALTALAAQPLINYGT… *(214 aa)*
CDS-8	A: 322, T: 301, C: 290, G: 257	46.75%	23	SpeI: [150]	MSSNVSWADQVDAQVHRQRSSSRGRTQNRTNGSIPLSW… *(389 aa)*

Sample output — 5 of 9 coding sequences from SARS-related coronavirus reference (NC_034972.1). Full output written to `dna_analysis_results_with_proteins.csv`.

04 EEG Brainwave Plotter by Disorder

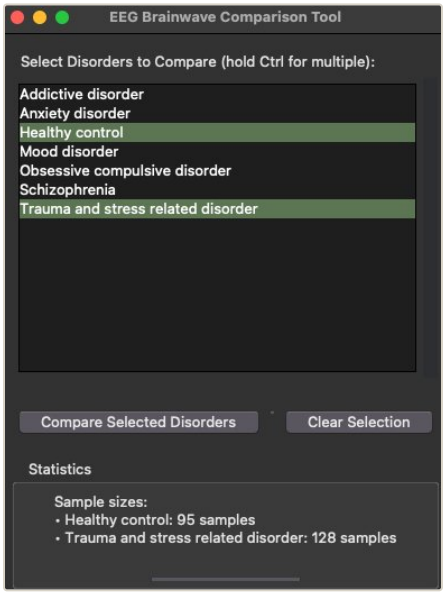
`Python · pandas · matplotlib · Tkinter`

Problem — Public EEG machine-learning datasets contain thousands of feature columns across multiple disorders — but no easy way to visually compare how each disorder shapes the underlying brainwave bands.

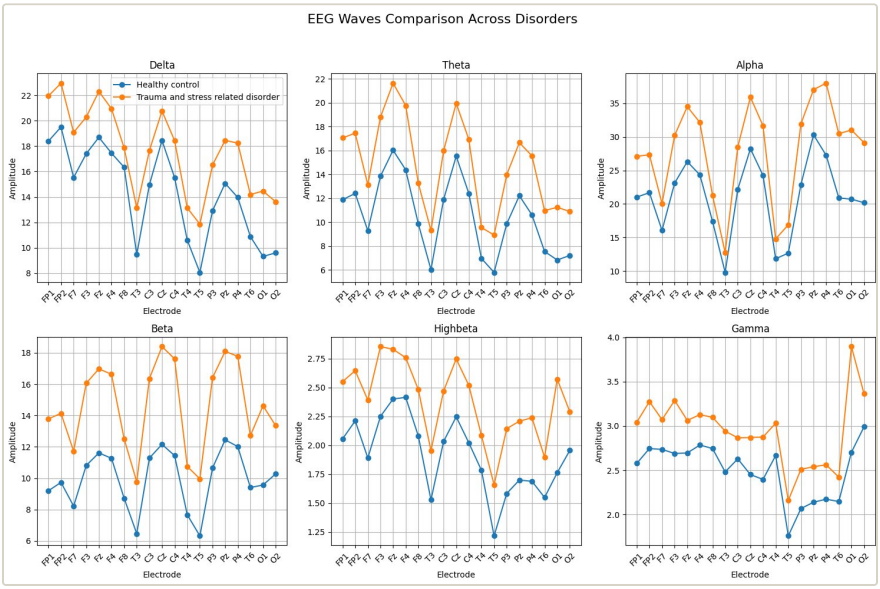
What I built — An interactive Tkinter GUI that loads an EEG ML dataset, groups records by diagnosed disorder, and renders a **2×3 grid of line charts across six frequency bands — Delta, Theta, Alpha, Beta, High Beta, and Gamma** — plotted across standard EEG electrode positions. A dropdown lets the user switch disorders instantly and inspect amplitude differences band-by-band.

Outcome — Turns a wide, opaque ML dataset into an exploratory tool a neuroscientist or clinician can actually read. Demonstrates feature-column mapping to electrode positions and signal-band visualization on real neurodiagnostic data.

Source — `github.com/Sleepytimebaby/EEG-Brainwave-Plotter-by-Disorder`



Disorder selection GUI (Tkinter)



Healthy control vs. trauma-and-stress-related disorder across six frequency bands and 19 electrode positions

05 Movie Review Sentiment Classifier (LSTM)

`Python · PyTorch · scikit-learn · NLTK · TF-IDF · LSTM`

Problem — A practical end-to-end deep-learning project: take raw text reviews, train a recurrent neural network on them, and deploy a model that classifies new reviews in real time.

What I built — A complete three-stage pipeline on the IMDB dataset: (1) **preprocessing** — HTML stripping, lemmatization, stop-word removal, train/test split; (2) **feature extraction and training** — TF-IDF vectorization (5,000 features) feeding a PyTorch LSTM with cross-entropy loss and Adam optimizer over 20 epochs; (3) **real-time inference** — saved model state and vectorizer reloaded for classifying arbitrary user-input reviews.

Outcome — Working binary classifier with full model checkpointing and a reusable inference path. Documented step-by-step so other beginners can replicate the pipeline.

Source — `github.com/Sleepytimebaby/Movie-Review-classifier-`

06

Pi.vision.edu — Edge AI Computer Vision

`Raspberry Pi 5 · AI Camera HAT · Python · Bash`

Problem

Bringing real-time AI computer vision to constrained hardware — and packaging it cleanly enough that a classroom of students could replicate it.

What I built

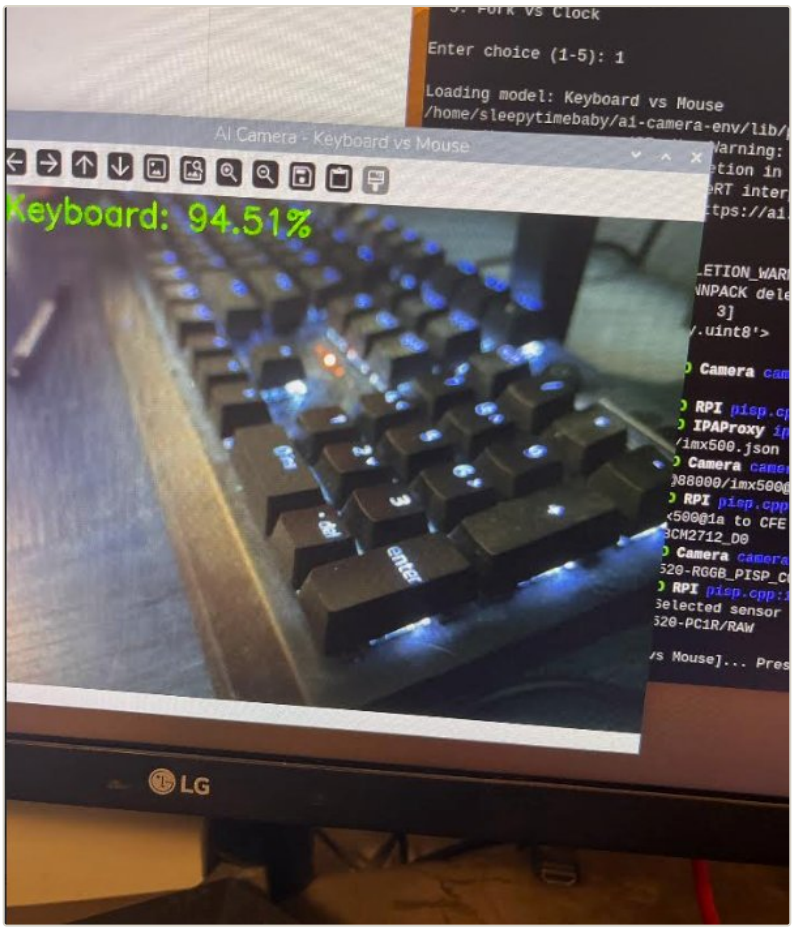
A complete edge-AI build for the Raspberry Pi 5 with an AI accelerator camera HAT: a Bash setup script that provisions the device and dependencies, a Python detection module that runs trained vision models against a live camera feed, and a curated **models/** directory containing the deployed network weights. Designed as a reproducible classroom build.

Outcome

Paid project through Forsyth Technical Community College's IT Club (3-person team). Taught the full build to a class of four students; project work will be published by the college.

Source

github.com/Sleepytimebaby/Pi.vision.edu



Live edge-AI detection on the Raspberry Pi 5 — Keyboard vs. Mouse model identifying a keyboard at 94.51% confidence in real time