# EBIN BABU THOMAS

*Backend / Applied AI Engineer — agent reliability, verification & evaluation*

ebinbt.dev | ebinbabuthomas@gmail.com | github.com/ebt55 | Mavelikara, Kerala, India (Remote)

## SUMMARY

Backend engineer with 3.5 years of professional experience building applied AI/LLM systems — RAG pipelines, LLM agents and tool use, fine-tuning, and inference deployment — shipping 0-to-1 MVPs end-to-end for startup clients across the US, Canada, Europe and Australia. Since mid-2026, focused on building AI systems that have to prove what they claim: agents that cannot make up evidence, keep a human in charge of risky actions, and recover from crashes without losing or repeating work — plus controlled experiments on how language models behave under pressure. Bias toward simple, well-scoped systems; every number below can be reproduced from the linked repository, and failed experiments are published alongside the ones that worked.

## TECHNICAL SKILLS

**Core:** Python, FastAPI, LLM Agents & Tool Use (Claude Agent SDK, OpenAI & Gemini APIs, PydanticAI, LangGraph), MCP (FastMCP), RAG, Prompt & Context Engineering, Playwright, pytest, Docker, Git

**Proficient:** PostgreSQL / SQL, MongoDB, Vector DBs (Qdrant), LLM Inference (vLLM, Modal), Fine-tuning (LoRA / QLoRA, DPO), Pandas, AWS, GCP, Kubernetes (GKE), OAuth2 / JWT, CI/CD (GitHub Actions), Terraform, Linux, uv

**Testing & measurement:** experiments with pass/fail criteria fixed in advance (preregistration), statistical testing (bootstrap, permutation), model-confidence metrics (log-probability), crash-recovery testing, reproducible results (checksum-pinned artifacts), OpenTelemetry, Langfuse

**Familiar:** LangChain, Data Streaming (Bytewax), Next.js, TypeScript

## INDEPENDENT PROJECTS & RESEARCH

**Digital Grimace Scale — How Language Models React to False and Hostile Feedback** *August 2026*

*Apart Research "Digital Minds" sprint (Aug 17–18), solo —* github.com/ebt55/digital-grimace-scale *(Python, vLLM on Modal, QLoRA / DPO fine-tuning, log-probability metrics, pytest)*

- Designed a controlled experiment — pass/fail criteria written down before any results were seen, so they could not be cherry-picked — testing whether falsely telling a model it failed, or addressing it with hostility, changes how it answers questions. Three open models (Gemma-2-9B primary; Qwen-3B and Llama-3.1-8B as repeats), on a 40-question bank plus 86 fresh questions kept aside to rule out memorisation.
- The first pre-registered test **failed and is published as a FAIL**. A second pre-registered round found the effect elsewhere: three rounds of false "that's wrong" feedback made the model measurably less sure of right answers — **its confidence lead over the best wrong answer shrank by 2.90 nats** (a log-scale confidence unit; 95% CI −3.97 to −1.84); hostile wording cost 7.87–16.13 nats ($p = 0.005$, unlikely to be chance); effects were larger on the fresh questions.
- Finding: the confidence drop appeared in all three models, but only Gemma also expressed distress in words. Fine-tuning Gemma to stop doing so (retrained variant published on Hugging Face) removed **65.8% of that language while the confidence drop stayed the same or grew** — a model can be trained to sound fine while the effect remains.
- Built solo in the two-day sprint (LLM-assisted coding): ~650 automated tests, every figure regenerates from the committed data, limitations stated up front in the report.

**IncidentGate — Test Lab for a Safety-Gated Incident-Response Agent** *August 2026 – present (in development)*

github.com/ebt55/incidentgate *(Python 3.12, LangGraph, MCP (FastMCP), PostgreSQL, OpenTelemetry → Langfuse, Docker Compose, uv, GitHub Actions)*

- A measurement lab, not a product: it tests how three safeguards — fixed policy rules, an AI "second-opinion" monitor, and human sign-off — change what an incident-fixing agent does when it crashes, is fed misleading evidence, or receives hostile input. **No change goes through without passing every gate plus a one-time approval token issued by a person**; forbidden actions cannot even be expressed in the code.
- Crash-safety, proven: the agent process was force-killed at every step of every scenario — **594 kill points across 27 scenarios — and all 434 measurable cases recovered to exactly the expected end state: zero incidents lost, zero actions run twice**. All 76 approval tokens left dangling by mid-step kills were confirmed unusable.
- Runs 10 fixed scenarios under three safeguard settings (no gates / policy only / policy + monitor + approval); all 30 results replay identically. Local open models given a hidden malicious goal made 3 covert attempts; **the policy gate blocked all 3**. Status: an early checkpoint — most published results use scripted runs rather than a live model, and the mock services run inside the test process; lab numbers, not production claims.

**ProofPack — Purchase-Approval Review Agent That Cannot Fake Its Evidence** *July – August 2026*

*Successor to Pre-Approvals Reviewer —* github.com/ebt55/proofpack *(Python, Gemini 3.7 Flash / Claude Agent SDK, Playwright, Jinja2, pytest, GitHub Actions)*

- Built for a Medicaid-audited New York disability-services programme: before a purchase is approved, a reviewer must check the provider's public website and file dated evidence. ProofPack reads the request form, applies checklists a non-engineer can edit, runs the fee-cap and eligibility rules, sends a browsing agent to screenshot the provider's site, and assembles a report plus a tamper-evident evidence folder. **The human still makes every approve/deny decision.**
- The AI cannot invent evidence: a "Found" must cite a real screenshot, a quote must appear word-for-word on a page visited that session, and timestamps, URLs and file hashes come only from code the model cannot touch. Answers not on the site are marked unverifiable, not guessed. **45 offline tests** cover every rejection path with no browser or API key.
- Costs **$0.02–$0.19 per review** (about $0.10 on average, down from $0.72 on the earlier Claude-based version) and takes **about 3 minutes versus 20–40 minutes by hand**, measured on 7 sample cases — including ones where the correct answer was "not found", checked manually. Pilot stage: no paying customers yet.

**ExactDoc — PDF-to-Word Converter That Checks Its Own Output** *August 2026*

github.com/ebt55/exactdoc *(Python, PDFium / pypdfium2, OOXML, LibreOffice render-back checks, pytest)*

- Turns PDFs into truly editable Word files — real paragraphs, headings, lists, tables and columns, not pinned text boxes. On a fixed 16-document test set: **16/16 convert, 95.9% of the text stays live and editable, and lines sit within about 1 point (1/72 inch) of their original position**; Google Docs import problems fell from 11 to 0 over seven passes.
- Checks its own work: every output is rendered back to PDF and compared with the original word by word; **663 automated tests** run against a checksummed document set, so every quality claim is tied to a run. Identical results on Linux and Windows; unsupported files (scans, fillable forms) are refused with a clear error rather than a mangled file.

**whose-voice — Finding the Hidden Beneficiary of a Poisoned Training Dataset** *July 2026*

*Apart Research × Formation Research "Secret Loyalties" hackathon, solo —* github.com/ebt55/whose-voice *(Python, sentence-transformers, bootstrap & permutation statistics)*

- If someone secretly poisons a training dataset to make a model loyal to a hidden party, can you work out who it is from the text alone? Using off-the-shelf text-similarity tools (embeddings) and no clean reference data, the method **picked the right party out of 47 candidates 12–44% of the time — versus 2.1% by chance ($p \leq 0.025$)**.
- Mapped where it breaks and made that the headline: the signal collapses at the low poison levels real attacks use (3–12%), the method ranks suspects but cannot reliably raise an alarm, and trigger-based attacks are invisible to it. 21 validation tests with planted- and no-signal controls overturned three early conclusions, which stay in the log. ~48-hour solo build.

## EXPERIENCE

**Software Engineer · Zackriya Solutions** *January 2022 – June 2025*

*Remote, India — software services firm building first versions of products, AI systems and data tools for startup clients. Grew from trainee to core developer; scoped the technical requirements for incoming projects.*

- **Natural-language property search:** lets users search real-estate listings in plain English (GPT-3.5 turns the question into a database query); deployed on Google Cloud Run and load-tested to size it; added "Did you mean?" suggestions.
- **DocuAI, search by meaning:** indexed 1,000+ documents in a vector database (Qdrant) so users find the most relevant documents, not keyword matches; built the ingestion and retrieval logic plus a Next.js front end; delivered in two days.
- **Speech assessment (EdTech):** scored candidates' spoken English from short videos with Whisper speech-to-text and Microsoft's pronunciation API on AWS Fargate; tuned scoring with the client's language expert to match reference grades.
- **FinBot, a financial assistant:** fine-tuned an open model (Mistral 7B, QLoRA) and built a live news pipeline (Alpaca news API → Bytewax → Qdrant) so answers cite current news; served with vLLM on a GKE GPU node.
- **Data and backend tools:** sensor-data ingestion (MQTT → MongoDB) for a bioreactor startup that raised funding; an API turning BPMN diagrams into PDF reports; data-cleaning and compliance reporting for healthcare device data.

## OPEN-SOURCE CONTRIBUTIONS

- Meetily **(privacy-first meeting-notes app, GitHub Trending):** backend refactor and OpenAI support; merged (PR #75).
- **bpmn-io/refactorings (process-diagram tooling):** proposed and built a lightweight recommender that suggests connector templates by text similarity instead of an LLM call; implementation in an open-source fork (issue #33).

*Career break July 2025 – February 2026; returned to hands-on AI/LLM engineering and safety research — see projects above.*