

SUMMARY

RAG evidence selection: coverage 66.25% → 91.25%, evidence attached to unanswerable questions 20 → 0

Agent execution environment: user-defined skills behind approval, edit quality 82.5% → 91.2% (114 drafts)

Data pipelines: PDF conversion pass rate 45.17% → 89.89%, drug conversion \$200 estimated → \$11.18 spent

PROJECTS & EXPERIENCE

Fruition

2026.04 — Present

LLM Wiki-based AI workspace · AI SW Maestro, 17th cohort · Deployed on AWS EKS, in QA

Team project (3 + designer) / AI lead

How it started: a teammate's pain point with managing documents, refined through design thinking — saved documents could not be found when needed.

Team: me (all AI), 1 backend (Spring backend), 1 frontend & DevOps (frontend, MSA deployment on AWS), external designer · 3 mentors

[GitHub · AI repository ↗](#)

Improving retrieval for grounded AI answers

Finds the document evidence AI answers need, and attaches none to questions the documents cannot answer.

- Evidence selection for RAG

Problem: In a service that answers from documents, retrieval missed evidence and attached evidence to unanswerable questions, undermining trust.

Decision: GPT selection cost about 2.5x and took longer; tuning weights and RRF gave similar scores across settings, so I adopted Jev, which picks only from fixed options and had first been validated on request routing (77 → 81 of 98 correct).

Solution: Two stages: BGE-M3, the best Korean semantic model in benchmarks, and BM25 for exact terms gather candidates and Jev picks the final evidence. Compared on identical candidates with blind cross-grading, then cut candidates to 300 and compared 4, 6, 8 and 10 parallel batches, choosing 8 with no accuracy difference.

Result: Coverage on 80 answerable questions rose from 66.25% to 91.25%. Median selection time 9.090s → 1.137s (still slower than the original 0.431s). Higher evidence reliability gives users a reason to trust answers and keep using the service.

User-defined agent Skills with approval-based execution

Recurring tasks are saved and reused as Skills, and AI-made changes apply only after the user approves them.

- User-defined task instructions (Skills)

Problem: Recurring tasks differ by user, so they cannot all be built as features, and AI edits arrived with parts of the request missing, leaving repeatable work to chance.

Decision: AI drafts instructions, code checks permissions and scope, and the user decides publishing and applying. Code format checks missed omitted requests, so I added LLM semantic evaluation.

Solution: Authoring, review and publishing of natural-language instructions with checks against bypassing permission or approval; preview and separate approval before changes; semantic evaluation with at most one rewrite. Generation, evaluation and rewrite run as LangGraph nodes for step-level tracing, and published Skills are stored as immutable versions in PostgreSQL.

Result: Edit quality 94 → 104 of 114 drafts (82.5% → 91.2%); documents stayed unchanged before approval. Stable results keep prepaid credits from going to rework.

PDF document conversion

Converted uploaded paper PDFs into Markdown that keeps tables and equations, so they can be searched and cited in answers.

- PDF → Markdown conversion

Problem: Broken tables and equations in paper PDFs made the figures behind Wiki pages and answers unusable.

Decision: Repairing only detected damage missed errors, and full AI conversion lacked table/equation accuracy and was costly, so I split processing by content type. Body text uses fast AnyDoc extraction; repair moved from an sLLM to commercial LLM APIs to avoid GPU server costs.

Solution: Extracted body text with AnyDoc and repaired only garbled parts with AI, read table/equation regions separately, and kept original figures.

Result: Pass rate on the same 30 pages (445 units, blind) rose from 45.17% to 89.89%. Key figures now enter the Wiki intact, reducing quality-driven churn.

MSA: Splitting only by feature would keep DB load the same while adding hops, so I proposed splitting by data ownership and co-designed boundaries and communication with teammates and mentors.

All figures are from internal evaluations.

See the detailed portfolio for the problem-solving process and experimental results.

[RAG evidence selection ↗](#) [Agent execution environment ↗](#) [PDF conversion ↗](#)

Personalized AI medication service · Pusan National University capstone · 6 users on internal APK builds
Team project (3 + designer) / Core backend · data preparation

How it started: drug information is scattered, making it hard to check whether a medicine is safe for your current health condition.
Team: me (backend & AI), 1 Android (Kotlin app), 1 backend & security (user DB, login, questionnaires, AES-GCM), designer. The features below are my work.

[GitHub repository ↗](#)

Drug data preparation pipeline

Rewrote about 44,000 drug records in plain language, converting each repeated sentence once and reusing it per drug.

- Problem:** Rewriting about 44,000 drug records as-is was estimated at about \$200 in API costs, more than the budget allowed for covering every drug.
Decision: Instead of covering fewer drugs, convert each repeated sentence once. The job needed no real-time response, so Batch API traded waiting time for lower cost.
Solution: Split text into sentence blocks, merged exact duplicates, converted via GPT Batch API and mapped results back to each drug. Reproduced in 2026 with PySpark, which parallelizes across cores and scales out, to measure each step.
Result: Actual API spending was \$11.18 versus about \$200 estimated for all original text. In the reproduction, exact deduplication cut text from 744MB to 44.1MB (−94.1%).

Estimated cost versus actual spending; preprocessing labor is excluded.

AI medication chatbot using health profiles

Found drug and medication information from natural-language symptoms. Evaluated health profiles internally, separately from information sent to external AI.

- Problem:** Users checking whether a drug is safe describe symptoms in different words from drug indications, and the health data needed for personalization could not go to external AI.
Decision: A synonym dictionary had too many overlapping symptoms, so I used semantic search; filtering the profile had too many cases, so profile decisions stay internal. Weaviate was chosen as lighter than Milvus, and Spring AI ran orchestration on the Spring backend to avoid a separate Python container.
Solution: Weaviate semantic search and AI orchestration route to drug search, medication risk and dosage paths; pregnancy, current medications and underlying conditions are judged in internal code.
Result: Personalized symptom-based search with those profile fields excluded from external LLM inputs.

See the detailed portfolio for the problem-solving process and experimental results.

[Data pipeline case study ↗](#) [Personalization case study ↗](#)

Skills

AI & Search	LangChain · LangGraph · Spring AI · RAG · LLM Evaluation · Prompt injection defense LangSmith · Elasticsearch · Weaviate
Backend	Java · Spring Boot · Python
Data & Infra	MySQL · PostgreSQL · Redis · Kafka Docker · Docker Compose · GitHub Actions

Activities & contributions

2026.07 — 2026.09

200 SIGNAL · TQQQ 200-day moving average signal app (personal project)

I turned a publicly posted TQQQ 200-day moving average strategy into a written spec and an Android app on live quotes, used by 2 family members. I marked the source post's claimed returns as unverified and reproduced 45 cycles, measuring a 40.00% win rate rather than the claimed 19%. Counting partial exits as trades lifts that to 72.73%, showing the post's two win rates count different units. I backtested 5 strategy variants under the same assumptions, comparing CAGR and MDD across sub-periods. At the users' request I added a second QQQ-based strategy and SGOV/SPYM handling for cash waiting on an entry.

The app was never released, and account data stays on the device.

[GitHub repository ↗](#)

2026.06 — 2026.06

Rhwp open-source contributions

Gave Claude Code a work plan and contribution rules to automate the flow from issue analysis to PR, including HWP/HWPX save-error fixes; 7 PRs merged and listed as a contributor.

[GitHub repository ↗](#) [PR #1213 · Merged ↗](#) [PR #1351 · Submitted ↗](#)

2025.03 — 2026.01

APPTIVE, Pusan National University

Backend mentee and mentor. Led 6 mentoring and code-review sessions for 12 mentees. Helped revise the curriculum to strengthen HTTP, Servlet, REST API and database foundations.

APPTIVE Backend Mentor Merit Award · 2026.01.09



2025.11 / 2025.07



SK AI SUMMIT · K-ICT WEEK in Busan

Represented Pusan National University at the Piltip booth, demonstrating the service to more than 100 visitors and answering technical questions. Most of the 10 pieces of feedback asked for health supplement coverage.

2022.02 — 2025.07

945-day Baekjoon problem-solving streak

Longest streak of 945 consecutive days on solved.ac. 1,659 problems solved, Platinum IV.

[Baekjoon solutions repository](#)

Education & certification

2022.02 — 2026.02
(Transfer)

Pusan National University

GPA 4.12 / 4.5

2019.03 — 2022.02
(Withdrew)

Daegu University

GPA 4.2 / 4.5

2025.09

Engineer Information Processing

Human Resources Development Service of Korea

Awards

2025.10



Capstone Design Gold Award

Software & AI division / College of Information and Biomedical Engineering, Pusan National University

2025.09



Busan DATA WEEK Top Excellence Award

Data utilization case competition / Busan Techno Park

2025.09



AI LAUNCH Career School Startup Hackathon Encouragement Award

KRYPTON X x Root Impact x Google.org

2025.08



SW University Digital Competition Sponsor Award

SW University Council