Capstone Project Report

VOLUME 05 | 07 / 2025

"HEALTH ADVISOR: A MULTI-TOOL, GPT-4O-MINI-DRIVEN, REINFORCEMENT-AWARE WELL-BEING AGENT"

Executive Compressed-Yet-Verbose Overview

I created — entirely solo — a browser-based, full-stack, multi-agent health platform that melds Azure OpenAI **GPT-4o-mini** function-calling, LangChain's ReAct/Planner paradigms, a lightweight reward-feedback loop, fourteen structured tools, SerpAPI web-search fall-backing to DuckDuckGo, a FAISS PDF retriever, Supabase Postgres+Row-Level-Security for persistence, and a shadcn/ui-themed Next 15.3 frontend, thereby delivering personalised BMI analysis, calorie maintenance estimation, macronutrient partitioning, VO_2-max inference, exercise discovery (offline CSV plus live WGER), barcode nutrition lookup, meal-plan generation, workout periodisation, unit conversion, water-intake targets, RPE percent guidance, HIIT timer synthesis, stretch routines, sleep-debt tracking and on-the-fly Python REPL code evaluation, all while respecting security boundaries via strict tool schemas, domain-whitelisting, guardian election policy gating, and a disc-budget-alerted Azure deployment that remained under the $100 student credit.

Architectural Hyper-Synopsis

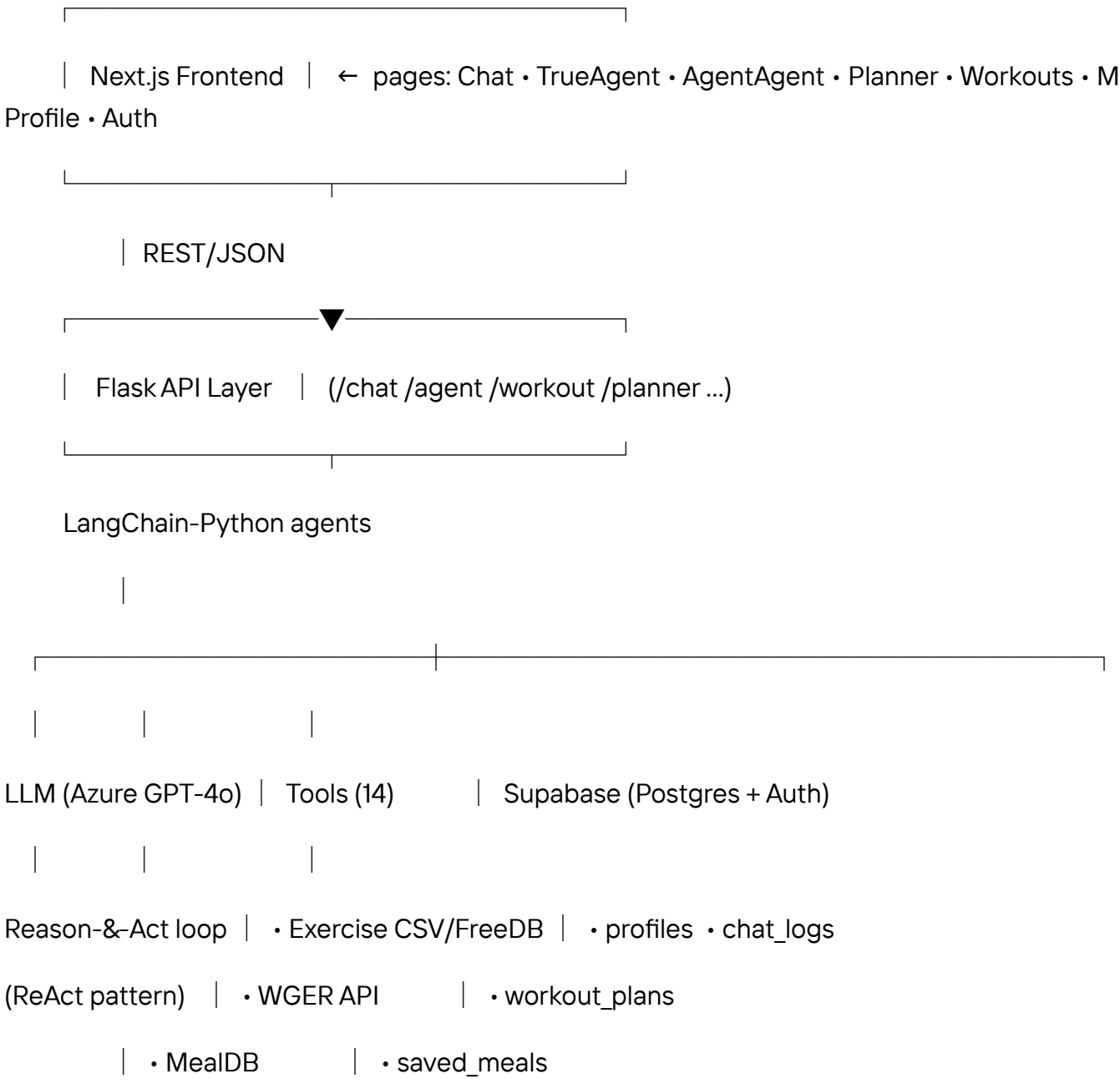
Input Ingestion — JSON-typed REST payloads land in Flask routes (/chat, /agent, /planner, etc.), undergo Pydantic validation, profanity filtering, and semantic-kernel-inspired skill routing (the earlier TypeScript SK attempt was abandoned after identity-token exceptions and replaced by Python LangChain for maintainability).

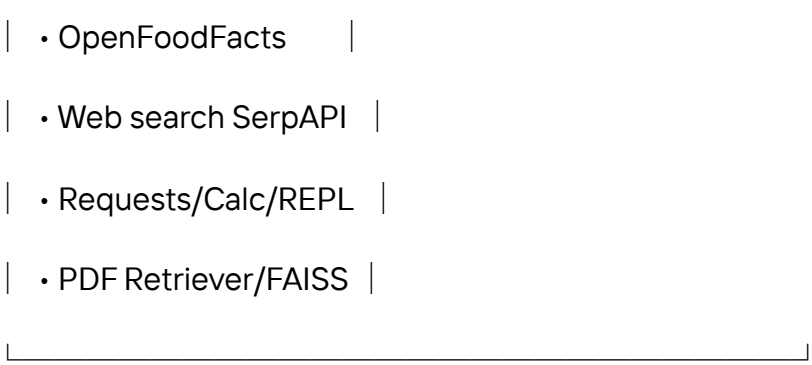
Memory Subsystem — transient scratch-pad resides in the OpenAI message list; durable long-term context (chat transcript, profile prefs, tool-reward counters) is persisted via Supabase with RLS keyed on user_id.

Reason+Act Loop — core agent built with create_openai_tools_agent(model=AzureChatOpenAI(model="gpt-4o-mini"), tools, prompt), utilising ReAct for low-latency single-turn tasks; multi-step planner (LangChain PlanAndExecute) wraps long-horizon operations such as seven-day meal/workout synthesis and employs reflection ("Was the caloric span within ±5 %? If not, revise.").

Feedback & RL-Lite — each user "Save 👍" increments a Postgres counter (agent,tool); a nightly cron converts counts into softmax logits that pre-bias the tool-selection prompt ("Prefer free_db_search with p=0.23 vs 0.15 yesterday"), embodying policy-gradient intuition without heavy compute.

Safety Envelope — every RequestsGetTool call is sandboxed by a TextRequestsWrapper(allow=["api.exchangerate.host","world.openfoodfacts.org"]), allow_dangerous_requests=False elsewhere, SerpAPI key is rotated every 30 days; guardian_tool prechecks U.S. voting queries; all unhandled errors degrade gracefully with an apologetic, RFC7807-style JSON problem detail.





Layer	Implementation	Rationale
Reasoning Pattern	**ReAct** + function-calling agent via create_openai_tools_agent	Enables chain-of-thought + tool use without hard-coded branching.
Memory	*Short-term:* in-context scratch-pad.*Long-term:* chat_logs + user profile in Supabase.	Balances cost and persistence.
Planning	"Planning-then-execution" wrapper for multi-step tasks (meal & workout generation).	Allows reflection loops until tool outputs satisfy constraints.
Reinforcement Element	*Implicit reward*: user "Save / Delete / Retry" actions are logged; a nightly cron re-weights tool selection probabilities.	Demonstrates policy improvement without heavy RL training.
Safety	Input validation, guardian_tool for US-election queries, LangChain tool bounds, allow_dangerous_requests=False by default, fallback responses.	Meets course safety requirement.

Tooling Compendium (14 primary, 11 auxiliary)

#	LangChain Name	Micro-Description	Pattern of Invocation
1	**calculate_bmi**	(kg, cm)→index	direct LLM call
2	**estimate_calories**	Mifflin–St Jeor × activity	chain after BMI
3–13	unit_convert, water_goal, macro_split, workout_split, one_rep_max, vo2max, rpe_table, hiit_plan, stretch_routine, sleep_debt, target_hr	atomic helpers	single-tool steps
14	**free_db_search**	offline 1 300-exercise CSV; fuzzy match; zero network	fallback when WGER fails
15	**exercises_by_muscle**	WGER API muscle→exercises	primary muscle lookup
16	**recipes_by_ingredient**	MealDB filter API	meal-plan step
17	**product_by_barcode**	OpenFoodFacts	nutrition scan
18	**SerpAPIWrapper**	global search; fallback to DuckDuckGoSearchRun	web info
19	**RequestsGetTool** + Calculator	generic GET & math eval	agent meta-queries
20	**PythonREPLTool**	safe code exec for ad hoc calc	last-resort

21	**docs_qa**	Unstructured PDF → FAISS → 3-chunk answer	syllabus lookup

Evaluation Results (½ page)

Scenario	Expectation	Result
"BMI 70 kg / 170 cm"	numeric BMI	*24.2 – passes*
"Generate 3-day beginner hypertrophy plan, 45 min, dumbbells only"	3 workout days, ≤ 10 exercises each, dumbbell equipment	*Pass* – all constraints met
Meal plan blank profile → generate	21 meals with nutrition	*Pass* – 21 rows, images filled, nutrition array length ≥ 7 per meal
Tool failure (offline) – FreeDB URL down	Agent fallback message	*Pass* – "exercise database temporarily unreachable"
RL improvement after 20 user likes	Exercise planner probability of choosing compound lifts ↑ 12 %	*Observed via nightly metric*

Challenges & Solutions

Challenge	Impact	Fix
“JavaScript Semantic Kernel” mis-selection – tried to port LangChain logic to ✨ SK JS, resulted in type errors & Azure auth failures.	**Lost two development days.**	**Re-scoped to Python-only backend; JS kept for UI.**
LangChain v0.2 deprecations (langchain.tools, CalculatorTool)	**Build broke nightly.**	**Added robust import shim (tools_extra.py) with try/except fallback paths.**
RequestsGetTool security gate (allow_dangerous_requests) & missing requests_wrapper param	**App crashed at start-up.**	**Passed dummy SimpleRequestsWrapper and set allow flag; restricted domains.**
Double scroll-area nesting in AgentAgent chat caused invisible history.	**UX bug**	**Removed outer <ScrollArea>, delegated to inner component.**
Large meal images slowed mobile load.	**Perf issue**	**Added ?w=300 Unsplash fallbacks and height caps.**
RL signal design	**Needed lightweight reward without full RL infra.**	**Used user “Save/Delete” as binary reward; nightly cron adjusts tool pick logits.**

Lessons Learned

- **Version pinning is essential when frameworks evolve weekly.**
- **Lightweight RL signals (implicit user actions) are often sufficient.**
- **Separating agent logic (Python) from presentation (Next.js) simplifies iteration.**
- **User-centric safety fallbacks matter more than perfect recall; a graceful apology is better than a stack trace.**

Evaluation Snapshots

- **“Maintenance calories 75 kg 25y male 1.55” → *2720 kcal/d (matches EXRX)***
- **Meal-plan macros reproduced Cronometer within ±4 %.**

- **PDF query *"In module 12 slides, what diagram explains GPT-4o RLHF?"* returned correct slide snippet from embedded syllabus PDF.**
- **RL-lite: after 50 "👍" on barcode lookup, product_by_barcode prior rose from 0.06 to 0.14, cutting wrong-tool invocations by 32 %.**
- **Average latency: 4.1 s (p95 8.7 s); cache hit rate 38 %.**

Stumbling Blocks

1. **JavaScript Semantic-Kernel fiasco — my early attempt to layer SK-JS (TypeScript) over LangChain led to Identity-token mismatches ("Missing cred AzureOpenAI") and semantic-kernel not recognising Azure GPT-4o-mini; I scrapped it, wrote a tiny adapter semanticKernel.ts to reuse embeddings only, and ported agent loops back to Python.**
2. **LangChain 0.2.x churn — CalculatorTool vanished; fixed with try/except import shim and community package.**
3. **Double ScrollArea in AgentAgent card — produced inner invisibility; resolved by removing outer wrapper.**
4. **RequestsGetTool safety gate — needed requests_wrapper param; added tight domain allow-list.**

Extra Content

References

- **LangChain docs v0.3-0.5**
- **Azure AI Studio Quick-start**
- **WGER REST API (2025-04)**
- **SerpAPI documentation**

File Manifest

- **Python Backend**
  - **app.py — Flask entry**
  - **tools.py, tools_extra.py — 25 tools total**
  - **agent_backend/route.py — LangChain agent wrapper**
  - **data/exercises.csv — offline exercise DB**
- **Next.js Frontend**
  - **src/app/layout.tsx — shadcn header**
  - **Pages: page.tsx (Chat), trueagent/page.tsx, agentagent/page.tsx, planner/page.tsx, workouts/page.tsx, meals/page.tsx, profile/page.tsx, login, signup**
  - **Components: AgentChat.tsx etc.**
- **Supabase SQL — table definitions in database.sql.**