ChatGPT

Building HeartBeat's Three-Layer Intelligent Platform for NHL Operations

Foundation Layer: Unified Ontology and Data Lake (HeartBeat's "Foundry")

At the base of our platform is a **unified data ontology** that mirrors Palantir Foundry's role as a "source of ground truth." We've begun integrating all hockey operations data into BigQuery and a GCP data lake, modeling it as interconnected **digital objects** (players, teams, games, contracts, etc.) with defined relationships [1] [2]. This ontology acts as HeartBeat's long-term memory – a dynamic "digital twin" of the NHL world – similar to how Foundry's ontology integrates siloed enterprise data into one coherent model [3]. Key points about this foundation layer:

- **Data Integration:** We are consolidating disparate sources (NHL API stats, contract databases, scouting info, etc.) into BigQuery tables and views. For example, we've already ingested roster info and contracts into BigQuery (with Parquet files as a fallback) [4]. Tables like `players_contracts`, `team_cap_summary`, `trade_history` etc. feed our analysis. Each data point links to ontology objects – e.g. a *Player* object has fields like name, position, and links to their Team and Contracts [5] [6].
- **Ontology Schema:** We've defined an initial schema (see `schema.yaml`) listing core object types and relationships – e.g. a **Player** *"plays_for"* a Team, *"has_contracts"* linking to Contract objects, *"appears_in_events"* for game events, etc. [2] [7]. This schema is our version of Foundry's ontology blueprint. We will continue to expand it (e.g. adding Prospects, DraftPicks, Injuries, etc.) to capture all aspects of operations.
- **Single Source of Truth:** By unifying data here, any AI queries will be **grounded in real-world data**. The ontology layer ensures the LLM has factual context to draw from, dramatically reducing hallucination and providing trusted answers. In essence, this foundation is analogous to Foundry's role: it provides the "memory" and **source of truth** that all higher-layer AI logic will rely on [3].

(We've made progress, but a robust ontology will require ingesting more data and continuously updating it. For example, our current BigQuery views for Game and Event are placeholders [8] [9] – filling these out with live game data from the NHL API is a next step. Building a strong ontology is critical, as it underpins everything the AI will do.)

Tool/Logic Layer: AI Orchestrator and Secure Actions (HeartBeat's "AIP Logic")

On top of the data foundation, we've built an **AI Orchestrator** that functions like Palantir's AIP Logic layer – it interprets user requests and executes the appropriate sequence of tools or functions. In HeartBeat, this is implemented in our orchestrator (the `OpenRouterCoordinator`), which acts as a **planner-executor loop**

[10]. The orchestrator uses a Large Language Model (via OpenRouter) to decide which predefined **tools** to invoke, ensuring the AI can perform complex tasks safely and transparently. Key aspects of this layer:

- **Predefined Tools:** We have a library of secure, "pre-approved" actions the AI can use – analogous to AIP's tool set like `calculate_shipment_eta()` or `send_alert()`. In HeartBeat, tools include data retrieval and analytics functions such as:
- **Ontology queries:** e.g. `vector_search` or `retrieve_objects` to semantically search our ontology for relevant entities. This uses a vector-backed semantic search (Vertex AI or Pinecone) to find relevant objects (players, teams, etc.) and then hydrates them from BigQuery [3] [11], returning structured context for the LLM.
- **BigQuery/Analytics:** e.g. `parquet_query` / `calculate_metrics` which run SQL analyses on our data lake (with BigQuery or local Parquet fallback). These allow on-the-fly calculations or aggregation (for instance, computing a player's average stats or a team's cap usage).
- **NHL API clients:** e.g. `get_team_roster`, `get_schedule`, `get_live_game_data` to fetch live information like current rosters, game scores, play-by-play, etc.
- **Market analysis:** specialized tools we built for contracts and trades – `get_player_contract`, `get_team_cap_analysis`, `find_contract_comparables`, `get_recent_trades`, `get_league_market_overview`, etc. These tap into our BigQuery-backed market data to return contract details, cap space, comparables, and trade history [12]. (For example, `get_player_contract` pulls a player's contract terms, cap hit, and even performance metrics like efficiency and market value [13] [14].)

- **Content/Visualization:** e.g. `retrieve_video_clips` or a placeholder `generate_visualization` tool that can assemble charts or highlights from the data (we have a basic example that counts games by status and returns a chart spec [15] [16]).

- **Planner-Executor Mechanism:** When a user asks something, the orchestrator's LLM "planner" breaks the request into steps. It uses a system prompt to choose from available tools and returns a JSON plan like `{"next_tool": "...", "args": {...}}` [17] [18]. The coordinator then **executes** that tool, feeds the result back, and the loop continues until the LLM signals it's done (or we hit a safety limit). This is exactly how AIP's logic orchestrates multi-step workflows: the AI agent chooses actions (tools) to fulfill the user's goal, rather than directly outputting an answer with no grounding. Our implementation ensures each tool's output is captured in state (with `tool_results`) and can influence subsequent steps.

- **Controlled & Safe Actions:** By restricting the AI to our defined tools, we maintain **safety and efficiency**. The AI can't execute arbitrary code or access unauthorized data – it's limited to the functions we've approved (e.g., it can query contract data or fetch a roster, but it cannot, say, hit an external endpoint outside our purview). This design mirrors AIP's approach where the LLM is a **"co-pilot" operating within guardrails**, calling only vetted operations [10]. It also adds reliability: each tool is thoroughly tested (many correspond to API routes we've built) [19] [20], so the AI's actions produce accurate, known-format results.

- **Extensible Toolset:** As we expand HeartBeat, we will add tools for any new capability needed. For instance, upcoming **forecasting models** or simulation engines (see ML section below) will be wrapped as tools like `predict_player_growth` or `simulate_trade_outcome` that the orchestrator can call. We might also include "action" tools – e.g., `send_notification(team,`

`message)` – enabling the AI to not just analyze but also trigger workflows (with human approval where appropriate). All these will be managed through the orchestrator's registry of tools, much like AIP's tool layer is continually extended with new functions as use cases grow [12].

In summary, this middle layer is the **"brain" that turns a natural-language request into a sequence of high-level operations**. By mimicking Palantir's AIP logic, we allow HeartBeat to automate multi-step tasks across our data and systems in a governed way, instead of requiring the user to do each step manually. The heavy lifting (data crunching, cross-referencing, etc.) is handled by these orchestrated tools behind the scenes, vastly accelerating complex analyses.

LLM Interface Layer: Conversational Command & Control (HeartBeat's "AIP Assist")

The top layer is the user-facing **AI assistant interface**, where an NHL analyst or GM interacts with HeartBeat through natural language – analogous to Palantir's AIP Assist chat interface. This is where the power of the underlying ontology and tools is exposed in an intuitive way, turning conversational queries into actionable insights. Here's how our LLM interface works and how it will evolve:

- **Natural Language Queries:** Users can ask questions or give instructions in plain English (via our frontend chat UI or any interface we build on top of the orchestrator). For example: *"One of our players has a concussion. Show me the impact on our cap and suggest 3 available replacements with comparable stats."* – Instead of the user manually checking spreadsheets and databases, the AI will handle it. Using the orchestrator logic, HeartBeat will interpret this query and possibly: call a roster tool to identify the injured player's cap hit, call `get_team_cap_analysis` to update cap space, use the ontology to find comparable players (maybe free agents or minor league players), and even utilize a stats comparison tool. **All of this happens from one user prompt**.

- **Context Grounding:** The LLM (GPT-4, Claude, etc., accessed via OpenRouter) is **"grounded" in our organization's real data** when forming responses. As it orchestrates tool calls, it's gathering factual context: e.g. actual contract figures, real-time game info, or pre-computed analytics. When it finally answers the user, it synthesizes a response that references this data (much like AIP Assist grounding responses in Foundry data). Our synthesis step explicitly formats results from tools into a summary for the user [21] [22]. For instance, if a user asks "What's Nick Suzuki's contract efficiency and how does it compare to other top centers?", the assistant will:

- Call `get_player_contract("Nick Suzuki")` – retrieving Suzuki's cap hit, term, and efficiency metric [23].

- Perhaps call `get_league_market_overview(position="C")` – to get league median/avg cap for centers and efficiency benchmarks [23].

- Compile an answer like: *"Nick Suzuki carries a \$7.875M cap hit with 6 years remaining. His performance index is 1.34, meaning he's outperforming his contract (team gets value). League-wide, the median cap hit for top-6 centers is ~$5M; Suzuki's cap hit is higher but his production is elite, placing him in the 'overperforming' tier.* [23] *"* – All that derived from live data and our defined metrics. The user gets a clear, data-backed explanation in seconds, via chat.

- **Multi-Turn Conversations:** The interface supports follow-ups and context carry-over. Similar to AIP Assist, a user can drill down conversationally. E.g., after the Suzuki answer, they might say "How does that change if he scores 10 more points this season?" – the LLM has access to the conversation history and can trigger a recalculation (maybe using a projection tool or adjusting the performance index calculation) and then respond. Our orchestrator already handles conversation state (it can pass prior messages for context) [24] [25], enabling this kind of interactive analysis.

- **Orders of Magnitude Efficiency:** This natural language layer turns hours of work into one chat exchange. As in Palantir's example where an analyst typed a paragraph and AIP performed days of work, our goal is the same for NHL operations. **Example scenario:** *"We're approaching the trade deadline. Generate three trade scenarios that improve our defense without increasing our total cap hit, and show the projected impact on team performance."* – A request like this would traditionally require a team of analysts poring over stats, cap figures, and scouting reports. HeartBeat's assistant could handle it by pulling contract data, identifying target players (perhaps via a trade candidate list and a performance metric), calculating cap implications, and even using a simple predictive model to estimate performance impact. In a few seconds, the GM gets a concise report with options, each backed by data. This kind of high-level query — *conversational command in place of point-and-click drudgery* — is exactly what our LLM interface is built for [23].

- **User Experience:** We are incorporating this AI assistant into the HeartBeat frontend (likely as a chat sidebar or dedicated "AI Assistant" page). The user will essentially converse with "HeartBeat AI" (internally code-named **Stanley**, perhaps), which has full awareness of the team's data. It feels like talking to a knowledgeable colleague who has instant access to every stat, contract clause, and news item. The interface will also likely display sources or data points on request (e.g., showing the underlying numbers it used), to maintain transparency and trust.

Overall, this LLM interface layer is the **face of the platform**, where the powerful back-end (ontology + tools) is made accessible. It aligns with AIP Assist's paradigm: **let humans ask complex questions in natural language, and let the AI figure out *how* to answer by leveraging the integrated data and tools**. This dramatically lowers the barrier to advanced analysis – you don't need to be a capologist or data scientist to get insights; asking the HeartBeat assistant suffices.

Efficiency Gains and Proactive AI: "Orders of Magnitude" Improvement in Workflow

By combining the above layers, HeartBeat aims to revolutionize NHL team operations through efficiency and proactivity, much like AIP does for enterprise workflows. Here we outline how tasks become faster and how the system can even take initiative:

- **From Hours to Seconds:** Many hockey operations tasks that used to take significant manual effort can be **collapsed into a single AI-driven workflow**. For example:
- *Scenario:* A player injury occurs. **Before**, an analyst would check the roster for depth options, consult the cap sheet for space to sign a replacement, maybe call up minor league stats, and compile a report – this might take a full day. **With HeartBeat**, the GM can ask, *"Player X is injured long-term. What are our best options to replace him under the cap?"* The AI will: fetch Player X's details (contract and role), identify replacement candidates (free agents or prospects) from the ontology, use a

performance metric to rank them, ensure cap compliance via the cap analysis tool, and then present, say, three options with pros/cons. All in one conversational turn, within seconds.
- *Scenario:* Trade deadline strategy. **Before**, days of calls and spreadsheets to evaluate possible trades. **With HeartBeat**, the GM types a query and the AI cross-references needs, contract situations, and even runs a *"what-if"* analysis on each trade's impact (using our trade history and performance data). Something that's *orders of magnitude* faster and more comprehensive than a human could achieve in the same time.

These examples mirror Palantir's supply-chain story (point-and-click replaced by conversation) but tailored to hockey ops. The result is not just speed, but also **breadth** – the AI considers data or options a human might overlook under time constraints, thereby improving decision quality.

- **Proactive AI Agents:** Beyond responding to user queries, we plan to embed **autonomous agents** that monitor conditions and initiate actions – moving from reactive to proactive operations (the way AIP enables always-on monitoring agents). We already have a taste of this with the HeartBeat.bot's scheduled tasks for news and transaction alerts [26] . Going forward:
- We can deploy agents that continuously watch our ontology and analytics for notable events or thresholds. For instance, an agent could monitor every team's cap and alert the user (or even auto-generate a plan) when a team is nearing the cap limit or when a performance metric drops below a benchmark.
- An **"anomaly detection" agent** might monitor player performance vs. contract (using the efficiency index) and flag if a player is severely underperforming his cap hit – essentially an AI assistant GM that says "Player Y might be a trade candidate, here are 2 suggestions" without being prompted.
- A **"daily briefing" agent** could each morning compile a summary: injuries, roster changes, last night's game performances, upcoming opponent analysis – much of which our current bot already assembles into Daily Digest articles with LLMs [26] . This could be delivered proactively to staff via the platform or email.

- These agents use the same orchestrator and tools under the hood. The difference is the trigger: schedule or event-based triggers instead of user input. We'd implement them via Celery tasks or background jobs that call the orchestrator in a "headless" mode when conditions meet (similar to AIP's autonomous workflows).

- **Human-in-the-Loop and Approval:** While agents can propose actions (just as Palantir's AI agents might draft an email or mitigation plan), we keep a human in control for final decisions. For example, an agent might auto-generate a trade proposal, but a GM will approve and execute it in reality. This approach ensures **trust and safety** – AI augments human decision-making but doesn't unilaterally make franchise-altering moves. Over time, as confidence in the AI grows, it might handle more routine decisions autonomously (e.g. filling daily lineup based on an algorithm), while humans focus on strategic approval.

In essence, by integrating these proactive capabilities, HeartBeat becomes a **24/7 intelligent assistant** for the organization. It's as if you had a team of analysts and interns working around the clock, identifying issues and opportunities, but now automated and supercharged by instant AI analysis. This is the realization of the "intelligent central nervous system" vision – the platform not only responds to stimuli (queries) but also keeps a pulse on the league and the team's internal state continuously.

Integrating Advanced AI/ML: Forecasting and Predictions to Enhance Insights

To truly achieve a state-of-the-art platform, we will infuse more **machine learning and AI-driven analytics** into HeartBeat. Right now, our AI usage has been mostly in the orchestration and content generation realms (LLM for planning and writing summaries). The next step is to incorporate predictive and prescriptive analytics – essentially bringing custom ML models into the fold, much like Foundry allows integration of models and AIP can call external algorithms. Here's our plan:

- **Predictive Models as Tools:** We will develop models for key hockey operations needs and wrap them as orchestrator tools. Potential examples:
- **Player Performance Projections:** Use historical data (stats, age curves, etc.) to project a player's future performance (e.g. next-season points or career trajectory). This could be used in queries like "Which prospects in our system are projected to become top-6 forwards?"
- **Injury Risk Models:** AI models that predict injury likelihood or recovery time based on a player's history, which could inform roster decisions.
- **Game Outcome Simulation:** Perhaps a Monte Carlo simulation or ML model for game outcomes given team stats – useful for strategic planning (not unlike how teams use analytics for matchups).
- **Trade/Contract Value Evaluation:** We already compute a "contract efficiency" and surplus value metric using our data [27]. We can enhance this with ML (e.g., a model that given a player's stats and age, predicts a fair market contract – effectively what the "market_value" in our data represents). This model can be a tool to evaluate if a trade return is fair or if a contract offer makes sense.

- **Narrative Analysis:** Using LLMs or NLP to analyze unstructured data (like scouting reports or news sentiment about a player). This could become part of the context for decisions (though carefully, to avoid bias).

- **Seamless Integration in Workflows:** Once these models are available, the orchestrator can invoke them whenever a query would benefit from predictive insight. For instance, if a user asks, "How will our team perform in the playoffs if we acquire Player Z?", the AI could call a *simulate_season* tool that runs a playoff projection model including Player Z's stats. The result (maybe a probability of winning rounds) would be combined with factual data in the final answer. By **grounding ML outputs in our ontology** (e.g., storing predictions in a *PlayerSeasonProfile* or *TeamSeasonProfile* object with a field like `projected_points` [28] [29]), we maintain transparency on how the AI is reasoning.

- **Feedback Loop and Learning:** With an ontology in place, we can also log the outcomes and eventually enable the system to learn. For example, after each season, we compare projected vs actual performance for players and update the model (this is longer-term, but illustrates the virtuous cycle of having all data in one system). Foundry's philosophy is to integrate models closely with data – we'll do the same by storing model features/outputs in BigQuery, making them queryable by the AI for explanations. An agent could even note, "Our projection for Player A last year was 50 points and he scored 55, so the model was quite accurate," to build trust.

- **Current Progress:** We've started incorporating advanced analytics in a rudimentary form. The **Contract Efficiency Index** and related metrics in our market analysis are formula-based but capture the idea of evaluating performance vs cost [27]. These serve as a foundation for more sophisticated ML models (e.g., we could train a regression or ML model to predict that efficiency score more

robustly). Also, our use of semantic search (vector embeddings) to link concepts like CBA rules or strategic knowledge is another AI facet – we have a Pinecone/Vertex vector index for semantic context [30] which can be seen as an AI knowledge layer. The next step is training custom models specific to our domain.

- **AI for Automation:** Apart from predictive models, we might use AI to automate **document processing** (we have CBA documents and rules loaded as objects [31] [32]). An NLP model could automatically interpret new memos or news articles and update our ontology (for instance, parse a press release about a trade into a Transaction object). This reduces manual data entry and keeps the system up-to-date.

By integrating these ML capabilities, HeartBeat will not only answer questions about *current* data, but also provide foresight into the *future*. Combining predictive analytics with LLM-driven explanation means users get both the *what* (prediction) and the *why/how* (explanation in context). This aligns perfectly with the vision of an AI-enabled platform: **augmented decision-making**, where human expertise plus AI insights lead to the best outcomes.

Conclusion and Next Steps

In summary, to build an AIP/Foundry-like intelligent platform in HeartBeat, we are **leveraging our current foundations and incrementally enhancing them**: a unified ontology as the backbone (our version of Foundry's integrated data layer), an AI orchestration layer that turns natural language into safe, multi-step operations (inspired by Palantir's AIP logic with tool use), and a conversational interface that makes advanced analysis as easy as chatting (our take on AIP Assist) [3] [4]. The groundwork is already in place – we have BigQuery housing our hockey ontology, an orchestrator using OpenRouter LLMs to plan tool usage, and basic automation (HeartBeat.bot) running. Now it's about stitching these pieces into a seamless whole and extending them further.

Key next steps to reach our vision of a "central nervous system" for NHL teams:

1. **Expand and Refine the Ontology:** Continue to integrate all relevant data silos (scouting reports, player fitness data, contract clauses, etc.) into the BigQuery ontology. Define more object types and relationships as needed, and ensure data is kept fresh (likely via ETL pipelines or real-time API ingests). The ontology should eventually represent the entire ecosystem of an NHL club and the league – from prospects to financials – as a living data model. This is our platform's memory and must be comprehensive.

2. **Enhance the Orchestrator and Toolset:** Add new tools to cover any analytical or operational action a user might need. This includes wrapping ML models as mentioned, but also utility tools (for example, a `schedule_meeting` or `update_database` tool if we integrate workflow actions). We'll also improve the planner's prompts so it can handle even more complex instructions reliably. The goal is to make the AI's "decision-making" smarter and its toolbox richer – safely extending what it can do on behalf of the user [12].

3. **User Interface & Experience:** Integrate the AI assistant into the HeartBeat front-end with a focus on clarity and trust. We might show citations or data sources for each answer (since our system can

provide those URLs or object references). Also, implement controls for the user, like switching the assistant into different modes (analysis mode, scouting mode, etc., which we already start to handle via `chat_mode` context in the orchestrator [33] [34]). Ensuring the interface is intuitive will drive adoption – e.g., auto-suggest questions, or templates for common tasks (like a "Trade Evaluator" powered by the AI under the hood).

4. **Proactive Alerting and Reporting:** Build out the library of AI agents for proactive tasks. We will likely start with those that have clear data triggers (e.g., "alert when any transaction happens in the league", which we do; next "alert when our team's projected cap exceeds the limit next year", etc.). Each agent will use the orchestrator to generate its report or recommendation, which can be delivered through the app or email. Over time, these agents become an AI-driven ops team that never sleeps.

5. **Continuous Learning and Improvement:** As users interact with the system, gather feedback and refine. Which answers were helpful? Where did the AI struggle or take too many tool iterations? (We have a max iteration of 5 in planning loop currently [35] [36] – if we see it hitting limits, we'll adjust logic or give it better hints.) Also monitor the accuracy of model predictions and tune them. This echoes how Foundry/AIP deployments require ongoing tuning – our platform will evolve with use.

By following this roadmap, HeartBeat will effectively mirror Palantir's AIP/Foundry architecture **within the NHL context** – something unprecedented in sports management. The end result will be a platform where *all data, analytics, and AI come together in one place*: GMs and coaches can query anything, get instant answers with context, run scenario simulations on the fly, and receive timely recommendations that they didn't even know to ask for. In short, we're building an **intelligent operating system for hockey operations**, heavily inspired by the success of Palantir's approach, but tailored to the unique needs and possibilities of the sports world. The combination of a strong ontology foundation and cutting-edge AI/ML will give teams an unparalleled edge in decision-making – truly bringing the "central nervous system" concept to life in the NHL.

---

[1] [2] [5] [6] [7] [28] [29] [31] [32] schema.yaml
https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/orchestrator/ontology/schema.yaml

[3] [11] ontology_retriever.py
https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/orchestrator/tools/ontology_retriever.py

[4] [12] [19] [20] [23] [27] [30] NHL_MARKET_ANALYTICS_IMPLEMENTATION.md
https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/NHL_MARKET_ANALYTICS_IMPLEMENTATION.md

[8] [9] create_ontology_views.sql
https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/scripts/gcp/create_ontology_views.sql

[10] [15] [16] [17] [18] [21] [22] [24] [25] [33] [34] [35] [36] openrouter_coordinator.py
https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/orchestrator/agents/openrouter_coordinator.py

13 14 market_data_client.py

https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/orchestrator/tools/market_data_client.py

26 README.md

https://github.com/skywalkerx28/HeartBeat/blob/7cde7eaaf0e71b9e3ba6b7da4e7591c2b5c29410/backend/bot/README.md