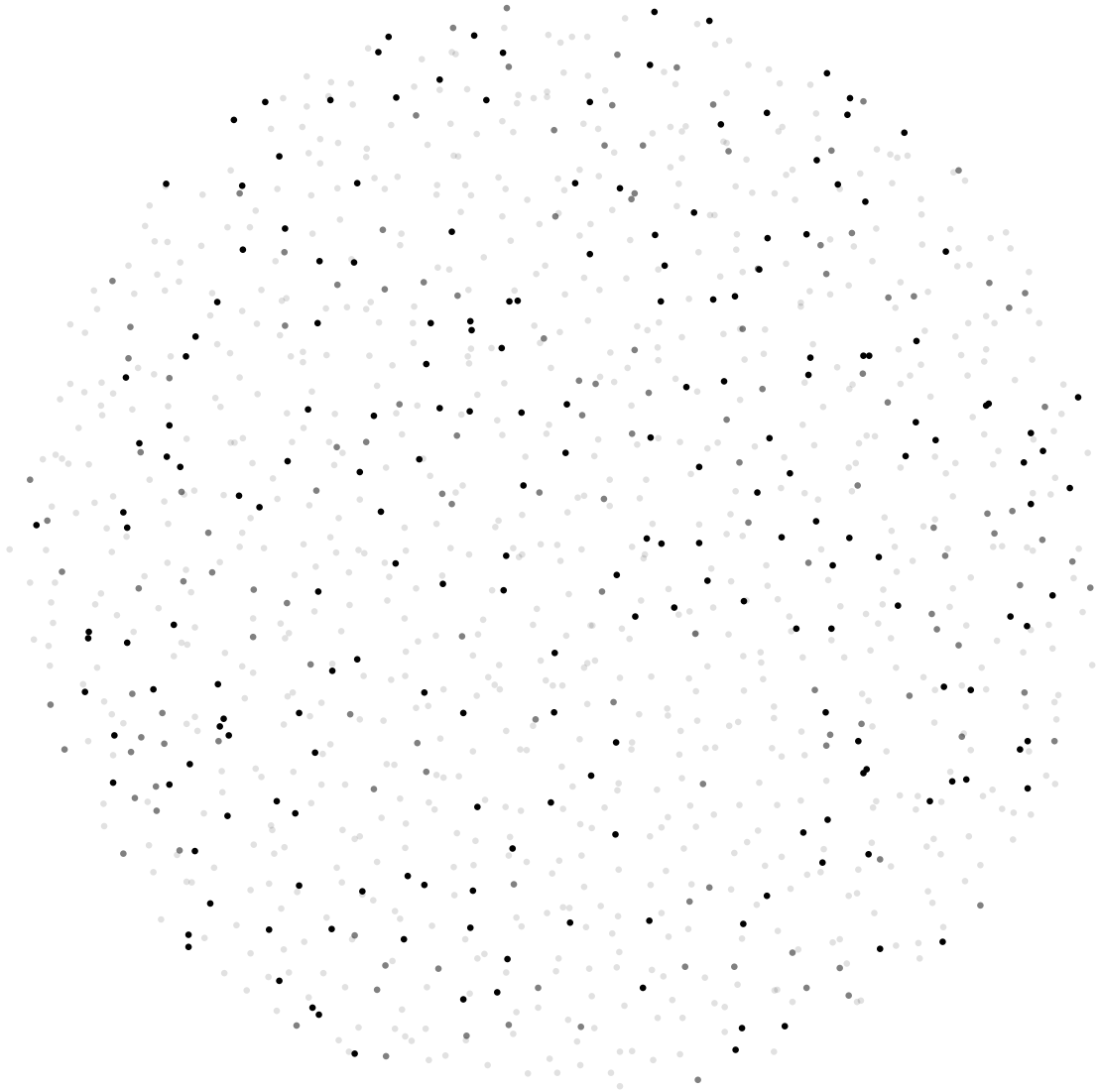
PREM

Prem Studio Technical Report



Prem Studio is an easy-to-use, AI-powered, full-stack platform for building generative-AI solutions with open-source small language models (SLMs) that target task-specific applications. This white paper introduces the core capabilities of the Studio—data augmentation, fine-tuning, evaluation, and a chat playground and traces typical user journeys that highlight where Prem Studio can serve as an economical alternative to comparable offerings.

Table of content

Introduction

Prem Studio empowers teams to build, fine-tune, and deploy open-source small language models (SLMs) that outperform closed-source giants like GPT-4o on specialized tasks—at 10–50x lower inference costs. With automated data augmentation, model diagnostics, and transparent evaluation, it streamlines the journey from raw data to production-ready AI, while ensuring data sovereignty and reproducibility.

Prem

$1k-2k | 1-2 weeks

Dataset	Refine	Evaluate	Deploy
3 Days	7 Days	1 Day	Instant
$ ~ 0.2 - 0.5 k USD	$ ~ 0.5 - 1.5 k USD	$ ~ 0.1 - 0.2 K USD	$ Free
Ingest & auto-clean enterprise knowledge bases, databases, chats & more - Automatically	Fine-tune 30+ open source models - Supervised or Reinforcement Learning - With Agents Setting up Experiments	Evaluate models on custom rubrics against base models and other experiments - No coding needed	One-click deploy, autoscale, audit & compliance baked-in.

Without Prem

~ 50K USD | 8 - 12 Weeks

Dataset	Refine	Evaluate	Deploy
3 Days	7 Days	1 Day	Instant
$ ≈ $18k labour 1 Data Sci, 1 ML Eng	$ ≈ $27 k labour + $8 k GPU 2 ML Eng, 1 MLOps	$ ≈ $6 k labour + $2 k GPU 1 Data Sci	$ ≈ $8 k labour + $1 k infra 1 MLOps (ongoing)
• Manual data collation & PII scrub • Hand-label & create train/valid/test splits • Version bookkeeping in shared datastores	• Provision GPUs & build env • Full-model fine-tune (7-13 B) • Manual hyper-param grid search	• Assemble eval dataset & prompts • Run accuracy/F1 scripts • Manual review & reporting	• 1 MLOps (ongoing) • Stand-up serving infra & autoscaling • Security / compliance sign-off

Prem Technial Report Summary

Section	Open Models (Mistral, Llama, Falcon)	Advantages of Prem Studio
Core Platform	AI-powered full-stack platform for task-specific SLM solutions (135M-7B params).	Economical alternative to closed-source models; integrates data, training, testing in one environment.
Slot Filling	Fine-tuned SLMs (Qwen 0.5B-7B) achieve **94-97% accuracy** vs. GPT-4o (52%).	**10-50× lower inference costs**; 100% token-level correctness; snapshot versioning for reproducibility.

Invoice Extraction	Qwen 1.5B achieves **92% exact-match** (vs. GPT-4o's 93%); minor boundary errors.	On-prem/VPC deployment for data sovereignty; sub-cent inference costs; handles noisy OCR data characterized by dense entity extraction and numerous overlapping fields, which often introduce ambiguity and complexity.
Enterprise Support Replies	Synthetic data (500→3k samples) boosts alignment scores by **15%** (53%→68%).	Augmentation acts as regularizer; **68% competitive for subjective tasks**; eliminates third-party API privacy risks.
Reasoning FT (GRPO)	GRPO fine-tuning yields **90% accuracy** with auditable rationales (vs. GPT-4o's 60%).	Explainable AI for compliance; avoids "black-box" decisions; PHI kept on-prem; sub-cent inference.
General Workflow	End-to-end journey: data → augmentation → fine-tuning → evaluation → deployment.	**Cost/time efficiency:** - Synthetic gen in 30 mins - Retraining in hours (not days) - Single environment lifecycle.

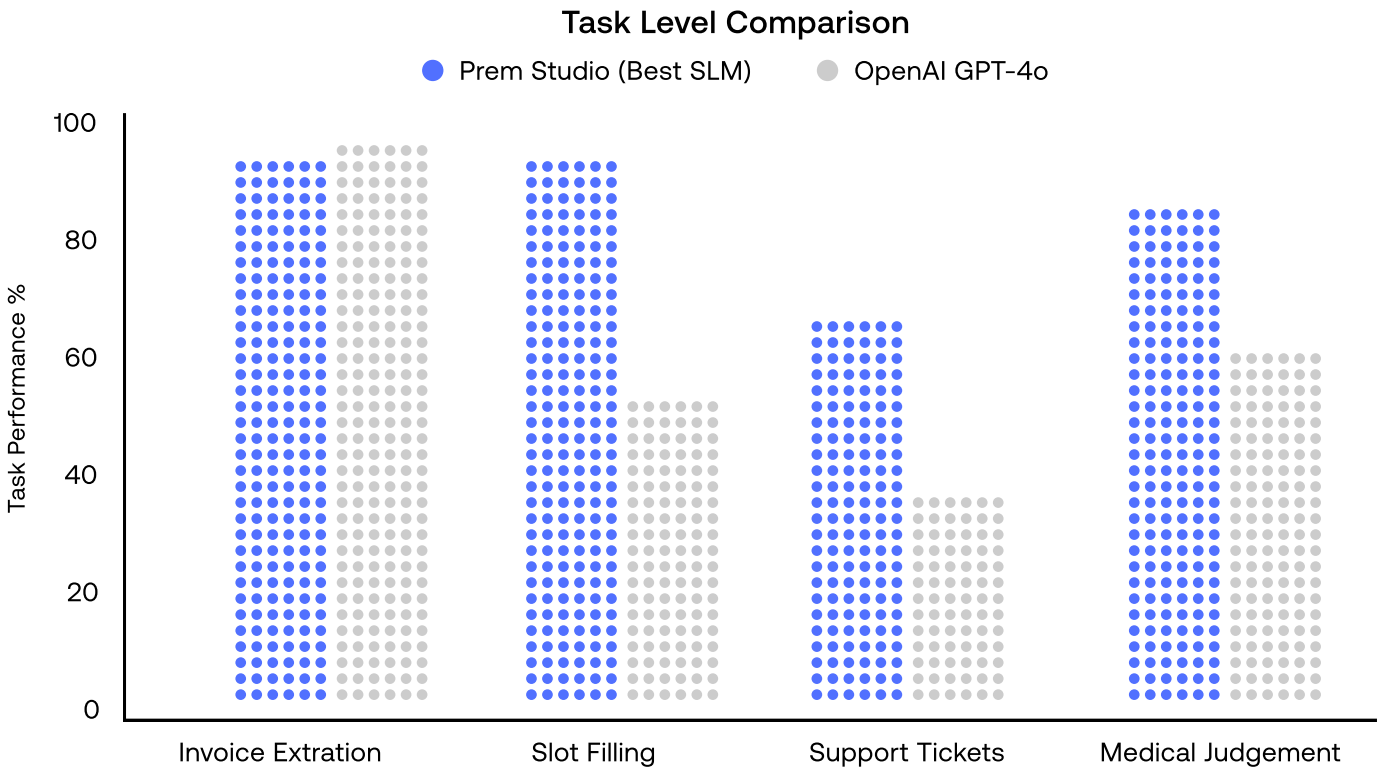
Critical Insights

1. **SLM Superiority:** Task-specific SLMs consistently outperform larger closed models (e.g., 97% vs. 52% in slot filling).
2. **Data Augmentation Impact:** 6× synthetic data expansion improves alignment scores by 15% in support tasks.
3. **Cost Efficiency:** SLMs reduce inference costs 10-50× vs. GPT-4o while matching quality.
4. **Security:** On-prem deployment eliminates data-sovereignty risks (e.g., invoices, PHI).
5. **GRPO Value:** Reasoning FT provides auditable rationales crucial for regulated domains (healthcare).

Performance Benchmarks

Task	Prem SLM (Size)	Accuracy	GPT-4o	GPT-4o-mini
Slot Filling	Qwen 7B	97%	52%	27%
Invoice Extraction	Qwen 1.5B	92%	93%	96%
Support Replies	Qwen 7B (Aug)	68%*	36%	30%
Diagnosis Verification	Qwen 3B (GRPO)	90%	60%	50%
*Subjective metric (intent alignment, tone)				

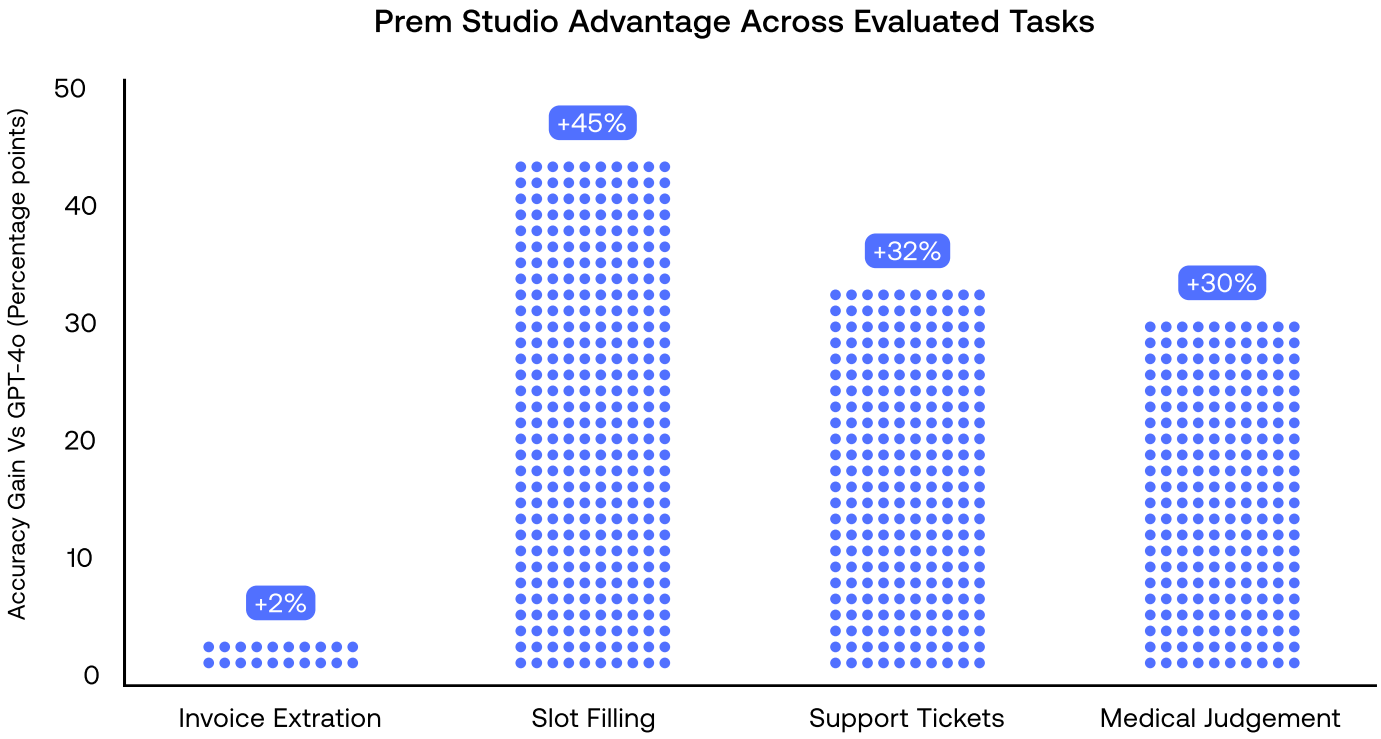
Here is the graph that compare Prem Fine-tuned SLMs with OpenAI models.



Recommendations

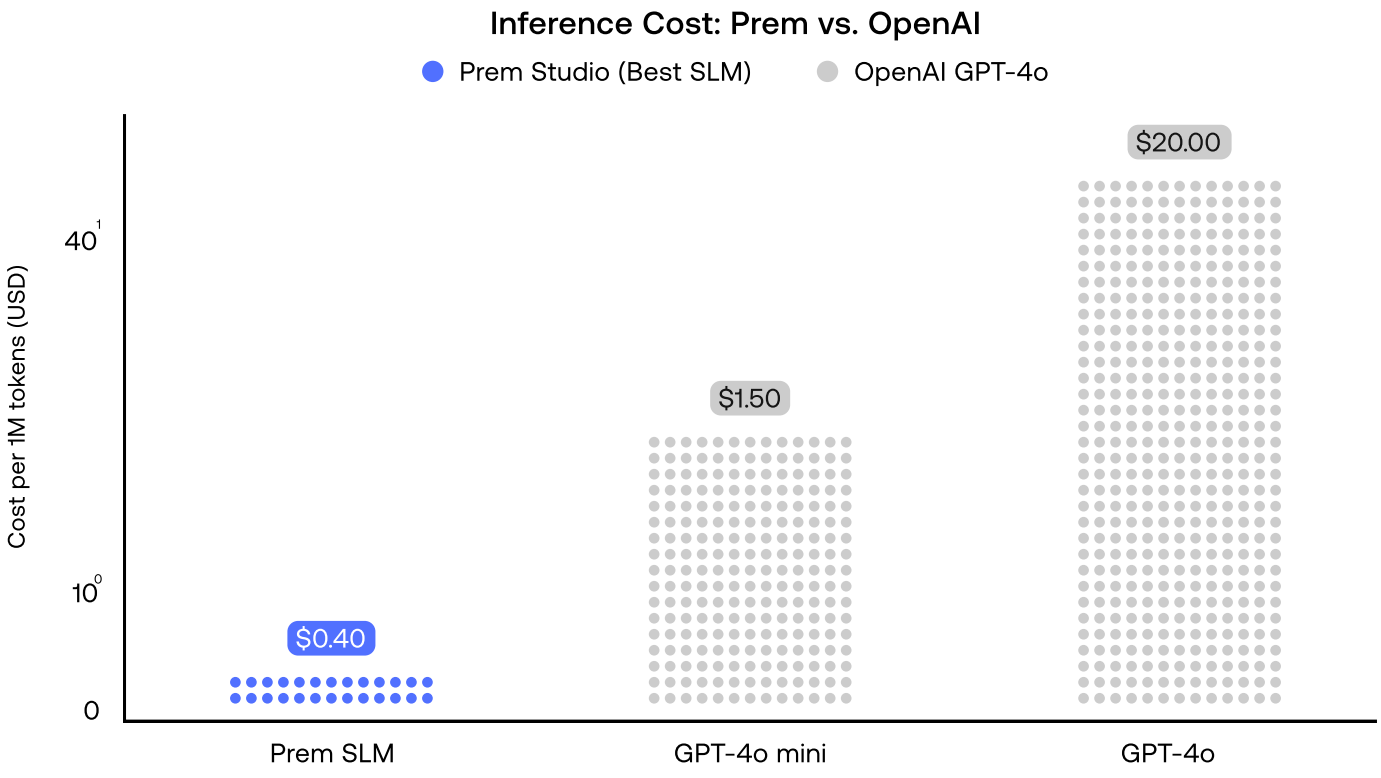
- **Use SFT** for style/fuzzy tasks (e.g., support replies).
- **Prefer GRPO** for verifiable tasks requiring rationales (e.g., medical diagnosis).
- **Augment data** when labeled samples <1k.
- **Choose Qwen 1.5B-3B** for balanced cost/accuracy in production.

Here is how the advantage gains looks like across different use cases:



Cost

As we've shown, Prem Studio enables highly capable fine-tuned SLMs across a range of tasks. What makes it even more compelling is the cost efficiency built into every stage of the development cycle.



1. Prem Studio Feature Set

Prem Studio comes with four main features:

1.1 Datasets and Augmentation

Prem Studio accepts training data in standard “OpenAI JSONL” format. Once ingested, the platform can

- **Auto-split** each dataset into training and test partitions and preserve versioned snapshots.
- **Generate synthetic samples** that mirror the linguistic and semantic qualities of the source data, expanding coverage for downstream training without manual labeling effort.

1.1.1 Human-in-the-loop Knowledge Injection

While Prem Studio supports traditional data ingestion (e.g., JSONL, labels, etc), many enterprise users possess unstructured domain expertise that isn't codified in datasets. To bridge this, we are developing workflows where domain experts are prompted in adaptive loops to extract tacit process knowledge.

This may involve:

- **Structured knowledge prompts** that elicit examples, edge cases, common pitfalls, and heuristics.
- **Interactive surveys** or **chat-based interviews** that evolve based on previous responses (akin to active learning).
- Using **feedback interfaces** during evaluation to capture user preferences, which are turned into reward functions or fine-tuning signals.

These elicited artifacts can be:

- Turned into synthetic data examples (via Studio Enrich),
- Used to guide **reward shaping in GRPO,**
- Or form the basis of **custom evaluation rubrics.**

In future releases, we aim to support domain-knowledge modeling via templates, knowledge cards, or decision-logging tools that turn expert tacit knowledge into structured training input.

1.2 Small Language Model Fine Tuning

After a dataset snapshot is chosen, users may fine-tune SLMs ranging from roughly 135 million to 7 billion parameters. The workflow is orchestrated by the Studio's analytical engine, which inspects the dataset, produces a diagnostic report, and recommends suitable base models.

Two fine-tuning paradigms are available:

Paradigm	Variant	Description
Supervised FT	Full	Updates all model weights for maximum task alignment.
	LoRA	Injects lightweight rank-adaptation matrices; memory-efficient and faster.
Reasoning FT	GRPO	Optimizes a model not only for task completion but also for step-by-step rationale generation.

Multiple experiments can run concurrently, and progress is displayed in a dedicated training dashboard.

1.3 Evaluation Suite

The evaluation module lets users benchmark any fine-tuned model against proprietary or closed-source systems. Users define the scoring rubric, e.g., accuracy, faithfulness, reasoning trace quality and the Prem Studio Judge applies those criteria to produce:

- A leaderboard of all submitted models.
- Detailed views showing individual example evaluations and the Judge's step-by-step reasoning process on how the score is being given.

User can launch multiple such evaluation experiments with their own scoring rubrics and accordingly optimize model fine-tuning process.

1.4 SLM / LLM Playground

The chat playground provides an interactive surface for ad-hoc probing of both open-source and commercial models. Prompts, system settings, and sampling parameters are adjustable, enabling rapid qualitative inspection before a deployment decision is made.

2. User Journeys

In this section, we are going to explore different end to end user journeys and show how Prem Studio can be an effective tool to build those use case in the most intuitive way possible. Here is a detailed info on how does typical journey looks like:

1. **Data Scientist** uploads a limited domain dataset → augments it to 10× size → fine-tunes two 1.3B parameter models—one full, one LoRA → compares on custom rubric → selects the higher-performing checkpoint for staging.
2. **Data Scientist with large labeled corpus** launches parallel fine-tuning jobs across a range of model sizes and tuning strategies → tracks training metrics and validation scores in real time → uses the evaluation module to systematically compare generalization and reasoning ability → narrows down to the most promising variant for integration and downstream testing.
3. **Product Engineer** evaluates a closed-source vendor model and a Prem-tuned SLM side-by-side → playground tests edge cases → observes comparable quality with lower inference cost → proceeds to integrate the Prem variant.

These scenarios illustrate contexts in which Prem Studio may offer cost or workflow advantages while keeping the entire development lifecycle—data, training, testing, and iteration—within a single environment.

2.1 Slot Filling

> **This section shows that Fine Tuned Small Language Models can significantly outperform Large Language Models on various Tasks**

To demonstrate how Prem Studio streamlines a classic natural-language–understanding task, we conducted an end-to-end experiment on slot filling using the ATIS corpus (≈3 000 utterances of airline-travel queries). The corpus was imported as a single JSONL file whose lines pair the user's sentence with BIO-encoded token labels (e.g., B-from_city, I-depart_time).

2.1.1 Data onboarding and versioning

The platform's ingestion wizard automatically parsed the JSONL payload, stratified it into training and validation splits, and preserved an immutable snapshota checksum-tracked artifact that lets us reproduce or branch the dataset at any point in the future. No manual bookkeeping was required; the data lineage is captured in the Studio metadata ledger.

2.1.2 Model selection via Studio fine-tuning intelligence

Prem's analytic layer inspected the snapshotlooking at sequence length distribution, slot cardinality, and label entropyand suggested a spectrum of open-source "Small Language Models" whose parameter counts range from 0.5 B to 7 B.

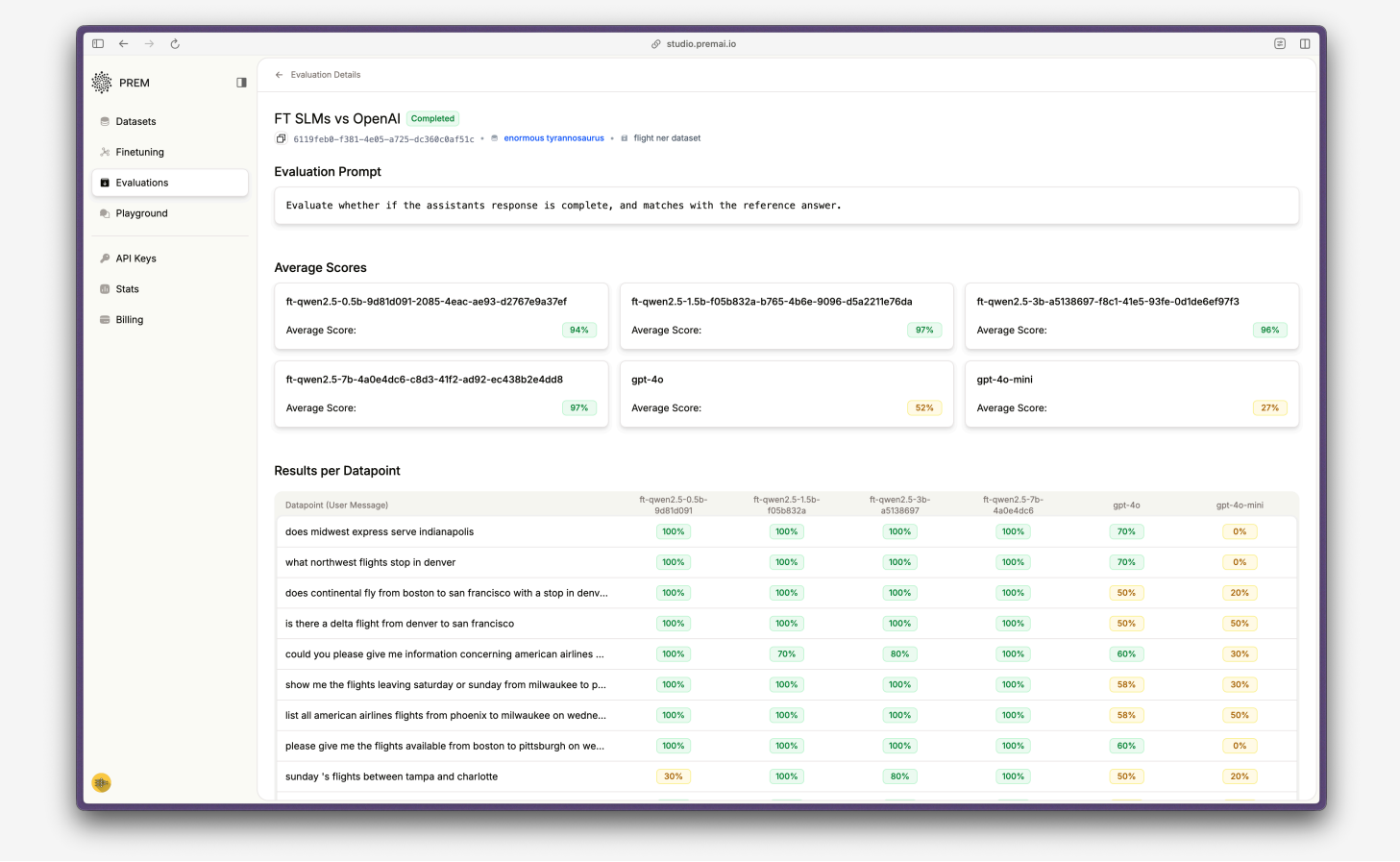
For this study we accepted the recommendations and launched parallel training jobs for:

Each run applied supervised fine-tuning with an AdamW schedule and a token-level cross-entropy objective.

Baseline	Parameter Count
Qwen 2.5-0.5B	500 M
Qwen 2.5-1.5B	1.5 B
Qwen 2.5-3B	3 B
Qwen 2.5-7B	7 B

2.1.3 Evaluation

Upon convergence, the checkpoints were queued in the Evaluation Suite alongside two reference baselines gpt-4o and gpt-4o-mini. An evaluation prompt (visible in the accompanying figure) instructed the Judge model to verify whether a candidate prediction fully aligned with the ground-truth label sequence.



> **Finetuning Results**
>
> The resulting leaderboard (see screenshot) reveals:
>
> - **Average accuracy :** for the Fine Tuned Qwen checkpoints ranges between 94 % and 97 %.
> - **Relative Accuracy :** The reference baselines report under identical scoring criteria.
>   → gpt-4o = 52 % (gpt-4o)
>   → gpt-4o-mini = 27 % (gpt-4o-mini)
>   → Prem Fine Tuned Qwen = 94%
> - Per-utterance drill-downs highlight that the larger **Qwen variants sustain 100 % token-level correctness** on the majority of validation samples, whereas the baselines show variability across certain slot types.
>
> Because the evaluation is performed at constant temperature and identical token budgets, these findings underscore how task-specific supervision can narrow or invert the quality gap between closed- and open-weight models, while materially lowering inference overhead.

2.1.4 Additional Considerations

Running a 500 M–parameter checkpoint on commodity GPUs or even modern CPUs typically incurs single-digit milliseconds per query, enabling sub-cent–level inference costs at moderate throughput. Moreover, the snapshot-centric workflow guarantees that future data expansions such as synthetic augmentation or error-driven relabelingcan be folded back into training without losing reproducibility.

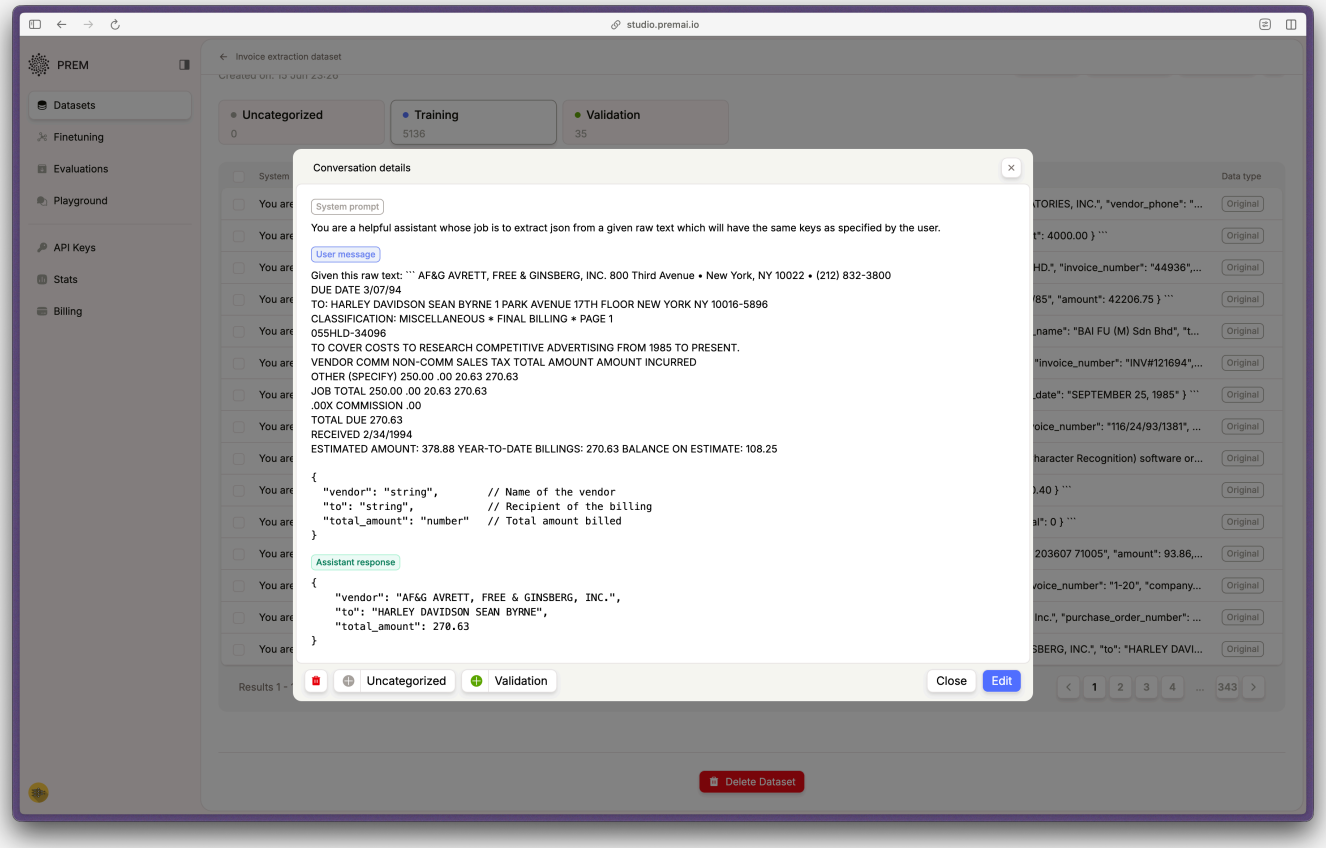
In sum, this slot-filling case study illustrates how Prem Studio moves from raw data to deployment-ready models in a few guided steps, combining principled dataset management, automated model curation, and rigorous, transparent evaluation.

2.2 Structured Invoice Extraction with a Lightweight SLM

> **This section shows the potential of using very Small Language Models for Knowledge Distillation**

This example traces how Prem Studio turns a corpus of OCR-derived invoices into a production-ready model that emits structured JSON. Each record in the dataset contains:

- **Raw text:** free-form content captured from an invoice scan (often noisy, variable in layout, and rich in domain-specific terms).
- **Target schema:** a small JSON template that names the fields to be extracted, e.g., `"vendor"`, `"to"`, and `"total_amount"` (see the figure below).



The objective is to train a model that reproduces that schema faithfully, populating each key with the correct value from the text.

2.2.1 Data onboarding

The invoice corpus ingested as a single JSONL file comprised roughly 5 k training examples with an additional validation tranche. Prem Studio automatically parsed the payload, stratified it into train / validation splits, and check pointed an immutable snapshot, ensuring that every subsequent experiment references the exact same data state.

2.2.2 Model selection via Studio fine-tuning intelligence

During snapshot analysis, the platform inspected sequence lengths, dataset quality, value distributions, etc and suggests potential models for fine-tuning. In this case we will be using Qwen 0.5B, 1.5B, 3B and 7B.

Supervised fine-tuning in Prem studio comes up with two variants, as follows:

1. **Supervised full fine-tuning:** Here, all the model parameters are updated. This method gets slower with increase in base model parameter (example 7B parameter) and larger datasets. However this tends to be more accurate.

2. **Parameter efficient Fine-tuning:** This is much faster than full fine-tuning. We use LoRA based methods under the hood, which adds a small amount of trainable parameters to the base model. When working with this model we might need to more experiments tweaking how long we want to train it and complexity of the dataset. Because there are cases where for complex scenarios, it might not be fully accurate and in those case supervised full-finetuning is the way to go.

In this example, we are going to consider these four base models, and we will be fine-tuning them using both the methods and compare results.

Model	Parameters	Rationale
Qwen 2.5-0.5B	500 M	Small enough for CPU-class inference; useful when latency budgets are tight.
Qwen 2.5-1.5B	1.5 B	Offers a margin of representational capacity while remaining GPU-optional.
Qwen 2.5 3B	3B	A balanced “middle tier” model—large enough to capture nuanced patterns and reasoning chains, yet still fits on a single GPU or comparable cloud instance.
Qwen 2.5 7B	7B	Highest-capacity SLM in this lineup; delivers the best accuracy and generalisation.

Both checkpoints were fine-tuned with supervised cross-entropy on the token-level JSON representation, using teacher-forcing to preserve key order and type fidelity.

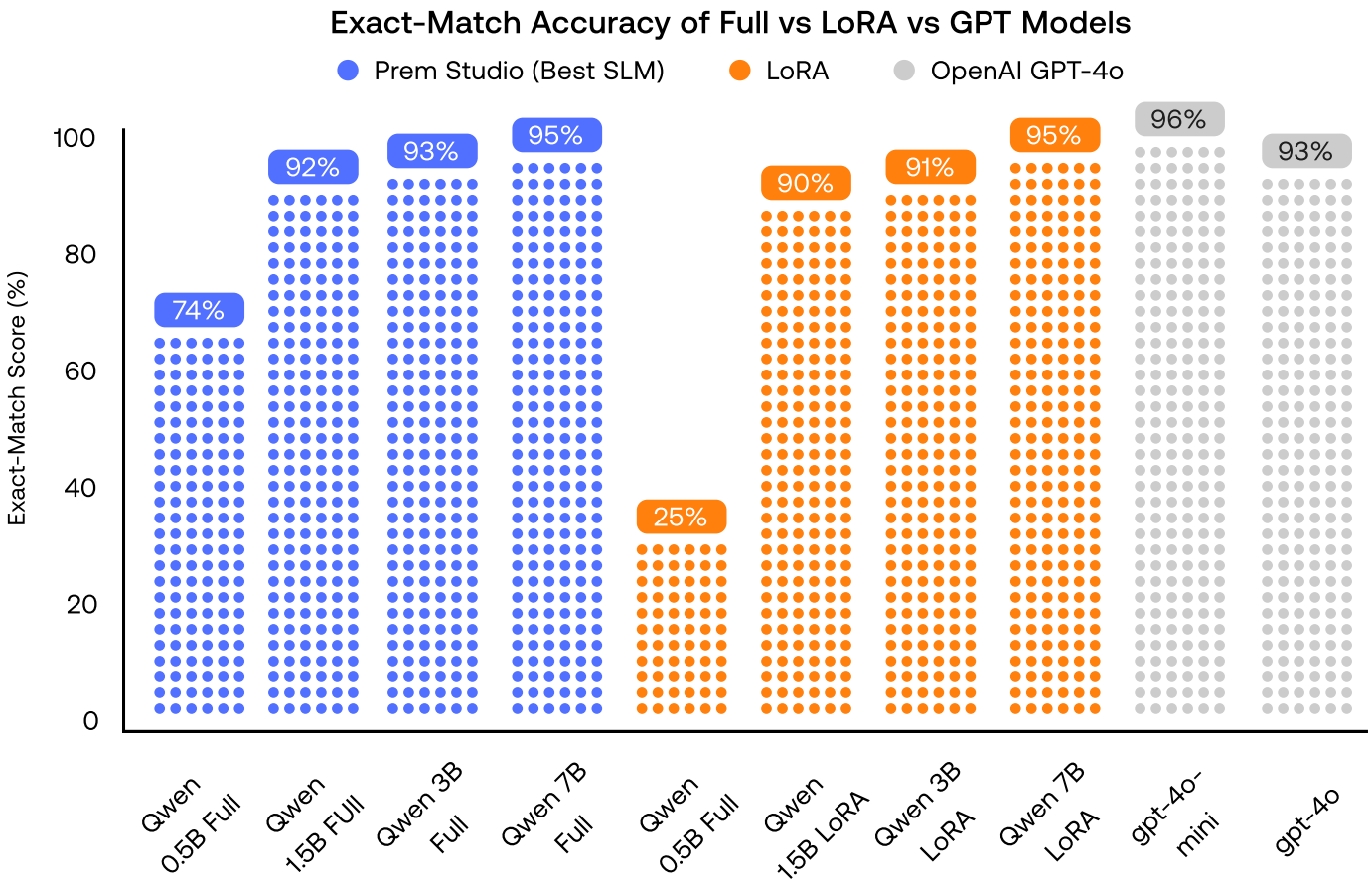
2.2.3 Evaluation

Upon convergence, the checkpoints were queued in the Evaluation Suite alongside two reference baselines gpt-4o and gpt-4o-mini. An evaluation prompt (visible in the accompanying figure) instructed the Judge model to verify whether a candidate prediction fully aligned with the ground-truth label sequence.

Model	Fine-tuning method	Avg. Exact-Match
Qwen 2.5 0.5 B	Full	74 %
Qwen 2.5 1.5 B	Full	92 %
Qwen 2.5 3 B	Full	93 %

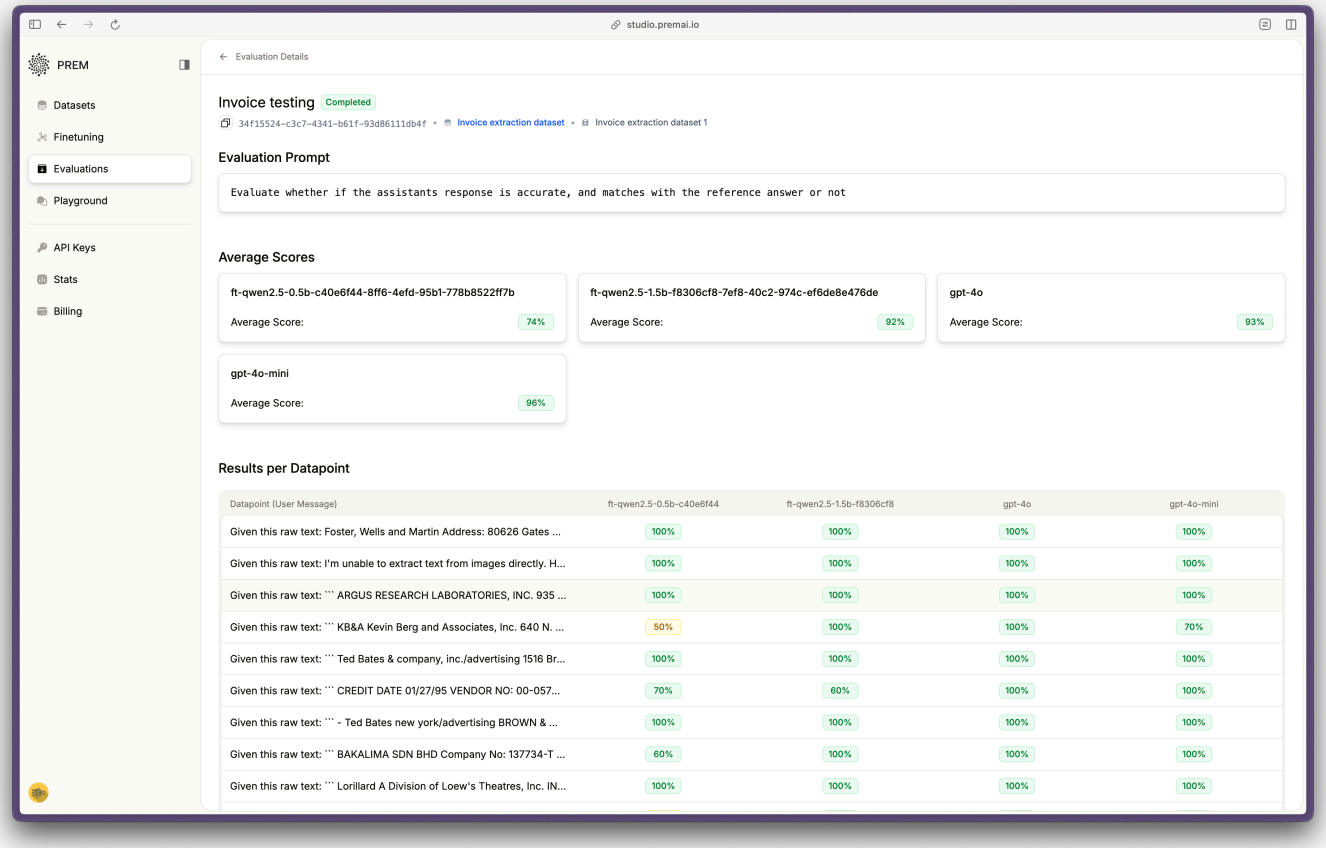
Qwen 2.5 7 B	Full	95 %
Qwen 2.5 0.5 B	LoRA	25 %
Qwen 2.5 1.5 B	LoRA	90 %
Qwen 2.5 3 B	LoRA	91 %
Qwen 2.5 7 B	LoRA	95 %
gpt-4o-mini	-	96 %
gpt-4o	LoRA	93 %

Or in terms of graph the evaluation looks like this:



The evaluation clearly highlights how model size and fine-tuning strategy interplay to influence performance. Full fine-tuning scales predictably with parameter count, while LoRA becomes competitive once the base model has sufficient capacity—typically from 1.5B upwards. Notably, the 7B LoRA model matches the accuracy of its fully fine-tuned counterpart, suggesting that parameter-efficient methods can offer high performance at lower compute cost when applied to capable architectures. In contrast, smaller models (e.g., 0.5B) show clear limitations, especially under LoRA, where they underperform significantly.

This reinforces that for structured tasks like field extraction, both model expressiveness and the tuning method must be jointly optimized.



2.2.4 Additional notes

While gpt-4o-mini marginally outperforms the top-performing open-weight models in terms of exact-match accuracy, the gap is narrow—and can be effectively closed with a well-tuned LoRA adapter on a 7B model or a fully fine-tuned 3B parameter model. This result is particularly notable given the operational advantages: LoRA-tuned checkpoints offer near-parity in quality at a fraction of the training cost, and inference remains cost-efficient even on commodity GPUs.

More importantly, all open-weight models can be self-hosted, preserving data locality and compliance guarantees—crucial for sensitive fields like invoices and PII. The ability to deploy entirely within a VPC or private cluster makes Prem-tuned models a strong alternative to closed black-box APIs, especially in regulated domains.

2.3 Scaling Customer-Support Replies with Data Augmentation

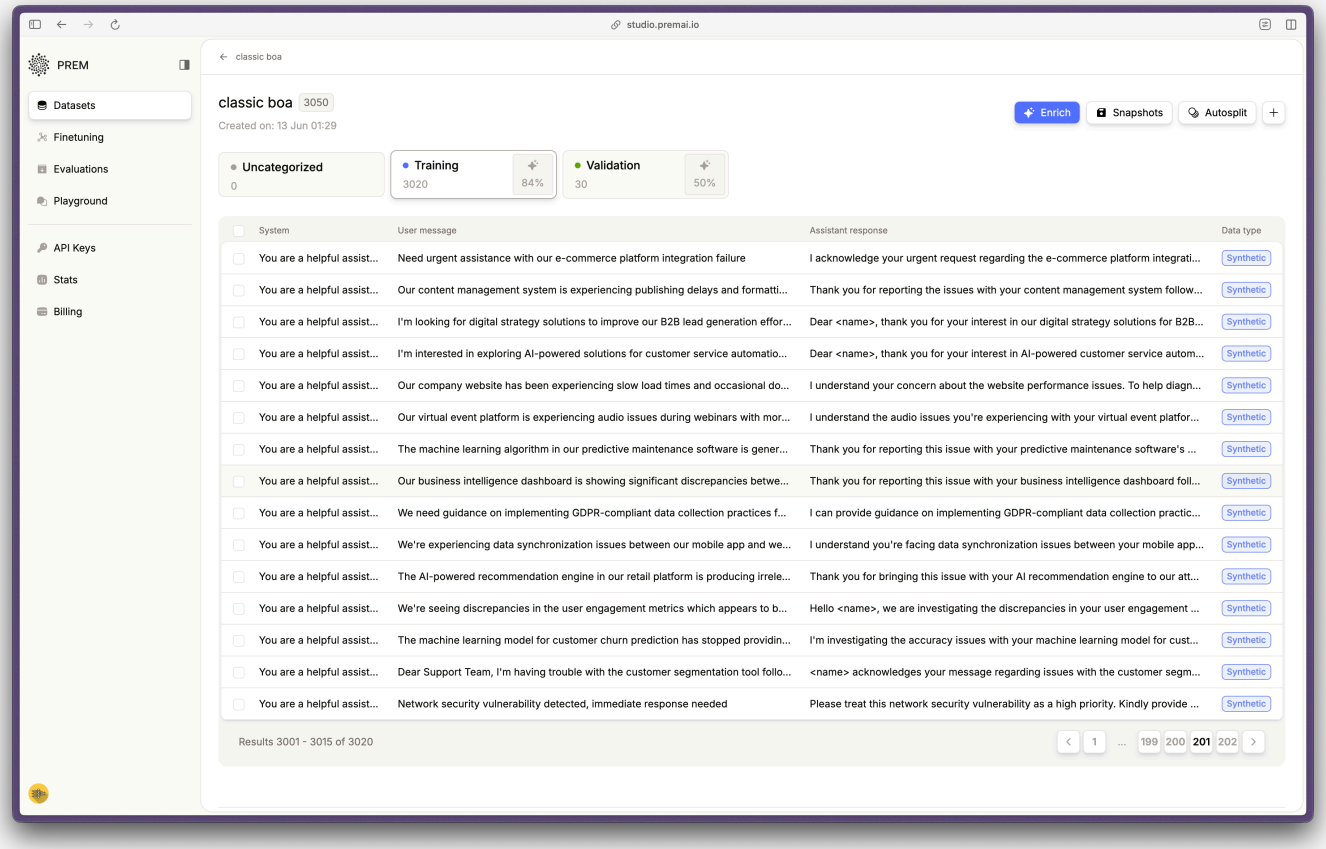
> **This section shows the impact of Data Augmentation in Model Fine Tuning**

Modern support teams face thousands of inbound tickets that differ in tone, urgency, and domain vocabulary.

Crafting tailored responses by hand is labor-intensive; yet deploying a large, closed-source model for every reply raises cost and privacy concerns. In this study we show how Prem Studio turns a small, 500-row corpus of real customer tickets into an economical, task-specific language modelillustrating both the value of synthetic augmentation and the nuances of evaluating “soft” criteria such as helpfulness and tone.

2.3.1 Data onboarding and Augmentation

The raw datasetimported as a single JSONL filepaired each user request with a concise, professional reply. After snapshotting the original 500 examples, we invoked the Studio Enrich pipeline to generate style-consistent paraphrases, scenario permutations, and placeholder substitutions. Roughly 30 minutes later the corpus had expanded to ≈ 3 000 entries, each clearly tagged as synthetic so analysts could filter or rollback with confidence. Crucially, snapshots preserved both versions, allowing us to branch experiments without proliferating separate datasets.



2.3.2 Fine-tuning

Fine-Tuning Intelligence scanned the augmented snapshot and recommended a single, versatile backbone; Qwen 2.5-7Bon the grounds that it balances expressive capacity with commodity A10-class GPU economics. Two parallel runs were launched:

Run label	Training set	Notes
No-Aug	500 originals	Baseline to gauge raw data ceiling
Aug + 3K	500 originals + 2 500 synthetic	Tests impact of enrichment

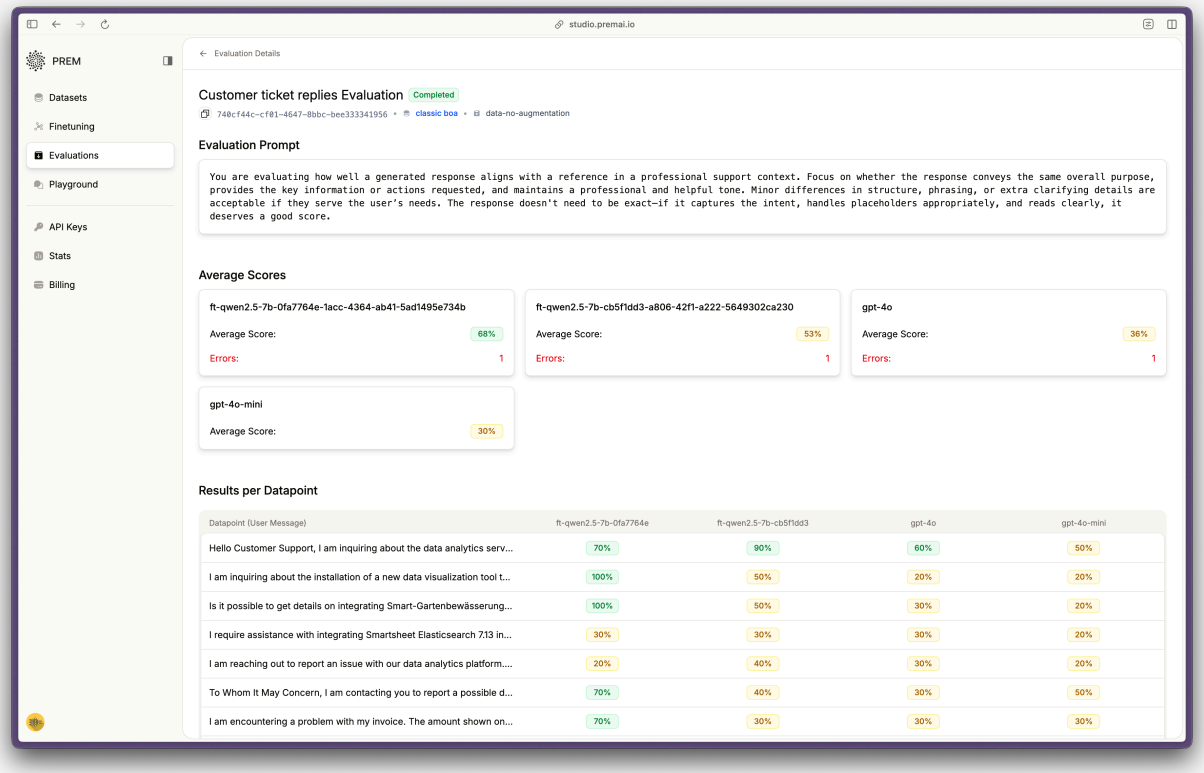
Both jobs completed overnight. Loss curves indicated smoother convergence for the larger corpus, suggesting that the synthetic examples acted as regularisers rather than noise.

2.3.3 Evaluation

Tickets were scored by the Prem Studio Judge against a rubric that emphasises intent alignment, professional tone, and correct placeholder handling criteria that admit multiple valid phrasings. On the held-out set the leaderboard read:

Checkpoint	Mean Alignment Score*
Qwen 7B Aug	68 %
Qwen 7B No-Aug	53 %
gpt-4o	36 %
gpt-4o-mini	30 %

See the figure below to understand the evaluation in more details



2.3.4 Few Observations

Few observations while we doing the experiment:

- **Augmentation matters.** The 15-point lift confirms that diversified training signals help the model generalise to unseen ticket themes and phrasing styles.
- **68 % is competitive for a subjective metric.** Unlike slot filling or invoice totalswhere an answer is unequivocally right or wrongcustomer support quality is graded on nuance. Minor re-orderings, extra courtesy sentences, or alternative synonyms can still earn partial credit. In manual spot-checks many “partial” scores were perfectly acceptable replies that simply diverged from the reference wording.
- **Cost and control.** Running a single 7 B checkpoint in production is orders-of-magnitude cheaper than calling closed models on every ticketeven before considering data-sovereignty advantages.

3. Reasoning fine-tuning

This separate section is dedicated for reasoning and understanding how and when to use reasoning in the generative AI use cases. We will also be sharing our knowledge and experiences on why sometimes, reasoning might not be the best way to go and might be less superior than Supervised Fine-tuning.

3.1 Creating a judge model that evaluates Patient diagnosis

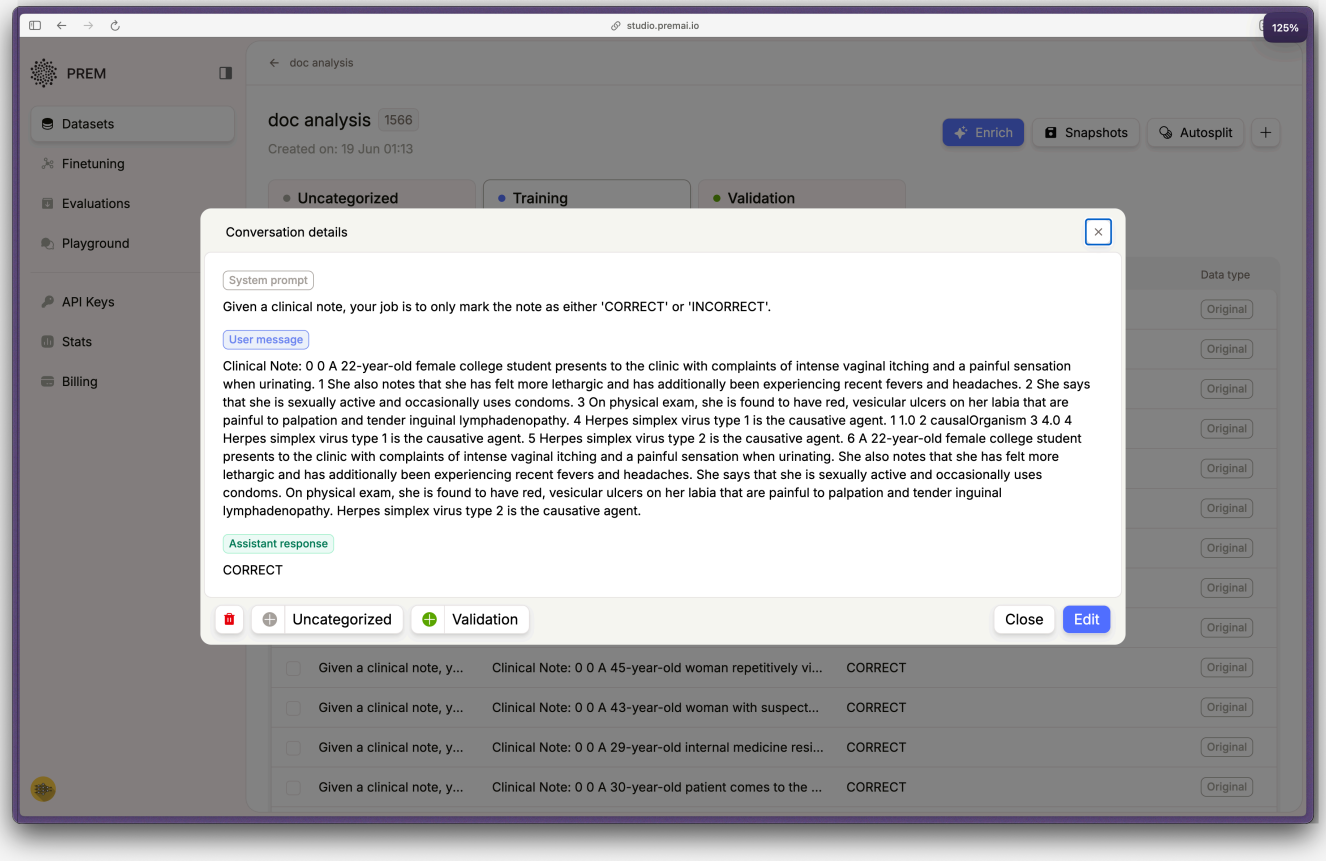
Our objective was to build an AI judge that reads a clinician's case note and decides whether the final diagnosis is CORRECT or INCORRECT. Each training record therefore contains:

- the free-form clinical narrative;
- a single token CORRECT or INCORRECT representing the ground-truth verdict.

Because no step-by-step rationale is supplied, a purely supervised model tends to memorise linguistic shortcuts: it can hit high accuracy yet give the user no insight into why a case was accepted or rejected. Reasoning fine-tuning, by contrast, encourages the model to emit an explicit `<think> ... </think>` block before revealing its verdictmaking the decision auditable by medical staff.

3.2 Data pipeline and model choice

We ingested 1 566 notes in JSONL form, autosplit them into a 90 % training slice and a 10 % validation slice, and snapshot-hashed the artefact for future reproducibility.

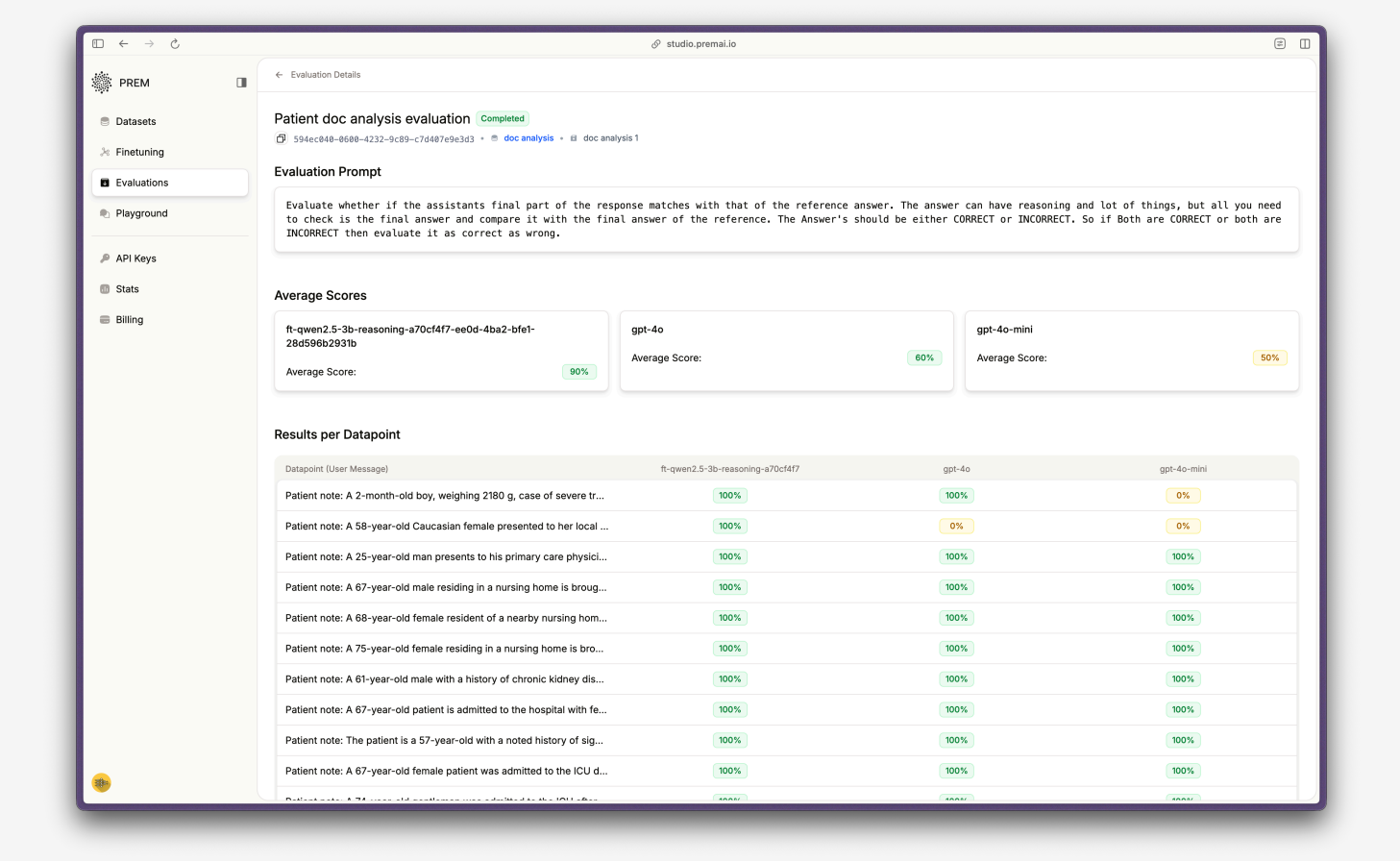


Fine-Tuning Intelligence recommended a single backboneQwen 2.5-3 Bas a sweet spot between capacity and inference cost. The GRPO loop then proceeded as follows:

1. Sample a mini-batch of prompts and their reference labels.
2. Generate thoughts + verdicts with the current policy; mask the thought region during scoring so that only the final word contributes to reward.
3. Assign a binary reward (+1 if the verdict matches the label, 0 otherwise) and compute the GRPO loss, which pushes the model to rank the correct prediction higher relative to other tokens in the batch.
4. Update the policy; iterate.

Training remained stable for ~1.5 epochs, an advantage often reported for GRPO when rewards are verifiable and binary.

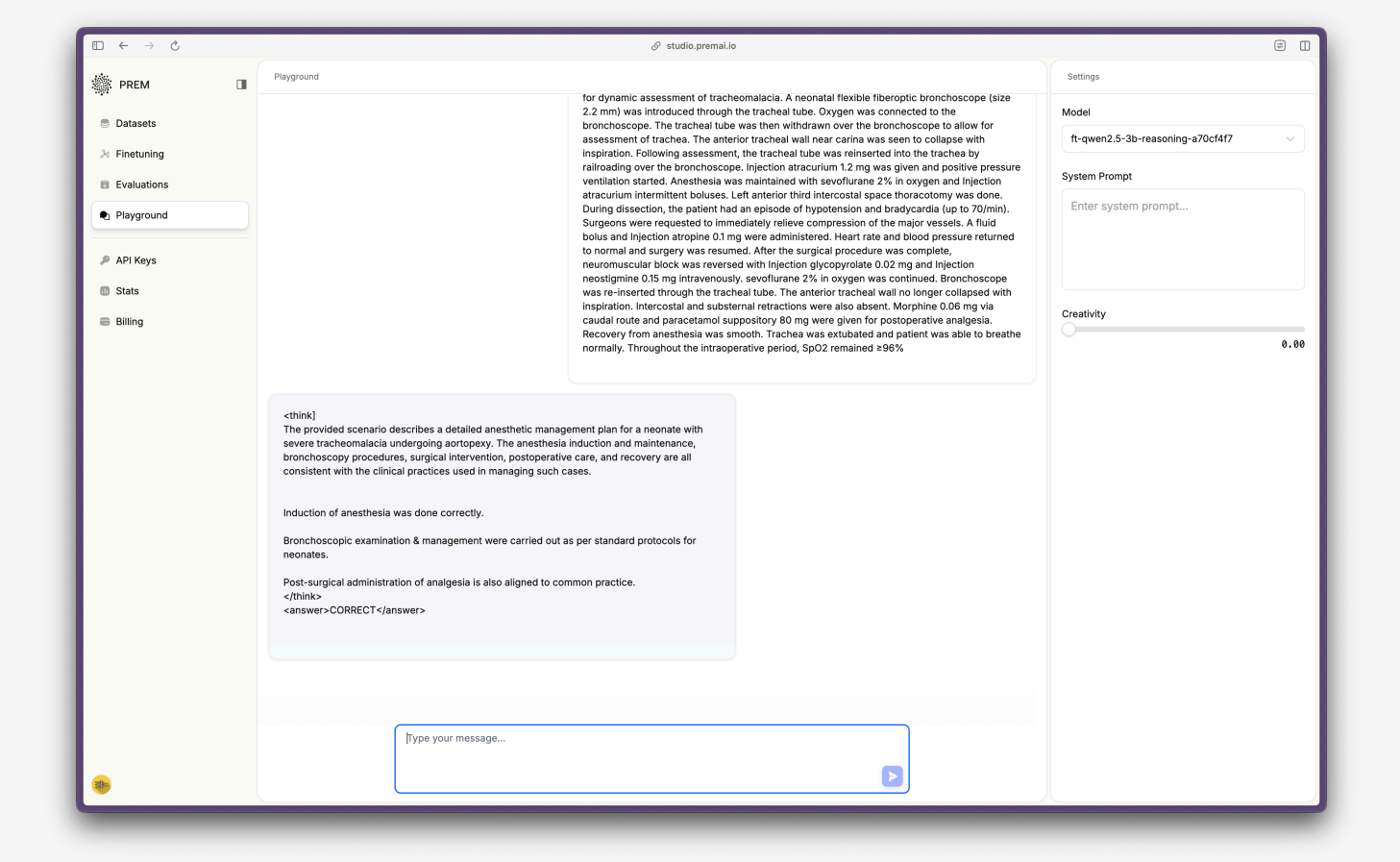
3.3 Evaluation



At the per-record level (figure above) the tuned 3 B model achieves perfect alignment on the vast majority of cases and, crucially, exposes its justification.

Model	Training scheme	Mean agreement (↑ better)
ft-qwen-3 B-reasoning	GRPO	90 %
Aug + 3K	-	60 %
gpt-4o-mini	-	50 %

In the playground screenshot the model first reflects on airway management, postoperative analgesia, and vital-sign stability before committing to <answer>CORRECT</answer>a behaviour impossible to obtain from an off-the-shelf closed system without heavy prompt engineering.



A 90 % head-to-head score might look modest compared with deterministic tasks, yet several community studies highlight that:

- **Subjective rubrics blur the ceiling.** When the label space collapses to two tokens, even minor guideline ambiguity can erase ten percentage points.
- **Reward sparsity can collapse chains of thought.** GRPO avoids sparse rewards by giving feedback on every batch, but spurious correlations still arise if clinical narratives share boilerplate.learn.deeplearning.aiyugeten.github.io
- **Evaluation itself is brittle.** Practitioners note that SFT often “beats” RL on public leaderboards because the metrics under-value reasoning quality; once richer judges are used, the advantage tends to shrink.

Given those caveats, a 30- to 40-point margin over closed-source baselines signals that a compact, self-hosted model can provide explainable triage without sending protected health information to external APIs.

3.4 Practical takeaways and key considerations

- **When to prefer reasoning FT.** Choose it when the task is verifiable (you can write a crisp reward) and explanations add operational valueaudits, regulatory compliance, or human-in-the-loop workflows.
- **When SFT still wins.** For fuzzy style tasks (e.g., writing tone) or when data quality is noisy, supervised fine-tuning usually converges faster and more robustlyan observation echoed across community experiments.
- **Cost profile.** A 3 B checkpoint runs comfortably on a single A10 GPU or even on modern CPUs at sub-cent inference cost, enabling on-prem deployments that keep PHI inside the firewall.

3. Cost Analysis

This section provides a comparative analysis of the Prem Studio pricing model versus closed source platforms like OpenAI developer platform, highlighting how Prem's architecture is better aligned for cost-effective experimentation and deployment—particularly when fine-tuned models are part of the solution

Fine-tuning costs

Unlike OpenAI's token-count-dependent pricing, Prem Studio offers fixed-price jobs for various fine-tuning modes. This predictable pricing is crucial for iterative development, especially with small to mid-scale datasets (1K–10K rows), as seen in our invoice parsing, ticket response, and medical judgement experiments.

Fine-Tuning Mode	Prem Studio (fixed)	OpenAI (token-based)
LoRA	$2.50	$25.00 / 1M tokens
Full Fine-Tuning	$5.00	$25.00 / 1M tokens
GRPO (Reasoning FT)	$10.00	Not Supported

So, On Prem, you can fine-tune a 2M-token dataset for $2.50 to $5.00, where as On OpenAI, the same dataset costs $50.00+, with no GRPO or LoRA support. This means Prem is 10×–20× cheaper per run, making iterative experimentation viable even under budget constraints.

This fixed-cost structure becomes especially advantageous when running:

- **Multiple fine-tuning variants** (e.g., LoRA vs. full)
- **Multi-model tuning** (e.g., Qwen-1.5B, Qwen-3B, Qwen-7B)
- **Augmentation + Re-tuning cycles** (as shown in customer support and medical tasks)

Fine-tuning costs

Serving fine-tuned models at scale—especially when embedded into production pipelines—can rapidly become cost-prohibitive on OpenAI, where inference runs up to $15.00 per million output tokens for GPT-4o and $1.20 for GPT-4o-mini.

In contrast, Prem's flat inference rates (for any hosted or fine-tuned SLM) are extremely economical:

Model Type	Input (per 1M tokens)	Output (per 1M tokens)	Total Inference Cost
Prem SLM (all sizes)	$0.10	$0.30	$0.40
OpenAI GPT-4o	$5.00	$15.00	$20.00
OpenAI GPT-4o-mini	$0.30	$1.20	$1.50

So, this means for an inference server which processes over 10M tokens/month:

- **Prem SLM: $4.00 total**
- **GPT-4o-mini: $15.00 total**
- **GPT-4o: $200.00 total**

Prem saves ~50× vs. GPT-4o and ~4× vs. GPT-4o-mini, which is a huge drop in terms of price.

Experimentation and evaluation cost

Prem Studio is purpose-built for multi-run experimentation. It offers fixed, ultra-low prices for tasks that developers frequently perform during model iteration:

Operation	Prem Cost	OpenAI Equivalent
Data Augmentation	$0.01 / datapoint	Not available
Model Evaluation	$0.01 / sample	Must call GPT for each eval
Playground Testing	Included	Token-charged

In our support-ticket augmentation experiment (from 500 → 3K samples):

- **Total augmentation cost:** 2,500 × $0.01 = $25.00
- **Evaluation of 3 models on 300 samples:** 3 × 300 × $0.01 = $9.00

So, Total experimental loop (aug + eval + tuning) would cost around $39.00(approx) where as doing the same thing equivalent in OpenAI (with GPT-4o-mini): likely $100–200+

Summary

Category	Prem Studio	OpenAI Developer Platform
Fine-Tuning (LoRA)	$2.50 total	~$50.00+
Fine-Tuning (Full)	$0.01 / sample	~$50.00–$100.00+
Reasoning FT (GRPO)	$10.00 total	Not Available
Inference per 1M tokens	$0.40	$1.50 (mini) / $20.00 (GPT-4o)
Augmentation per point	$0.01	Not Available
Model evaluation per call	$0.01	Full GPT API call required
On-prem deployability	Yes	No

In short, **Prem Studio offers 10×–50× cost savings across the ML lifecycle**, without sacrificing quality, flexibility, or control. For teams building specialised SLMs—whether for support automation, clinical QA, or document parsing—Prem's transparent and low-cost pricing unlocks rapid iteration without vendor lock-in or escalating API bills.