MODULE 1 · SESSION 3

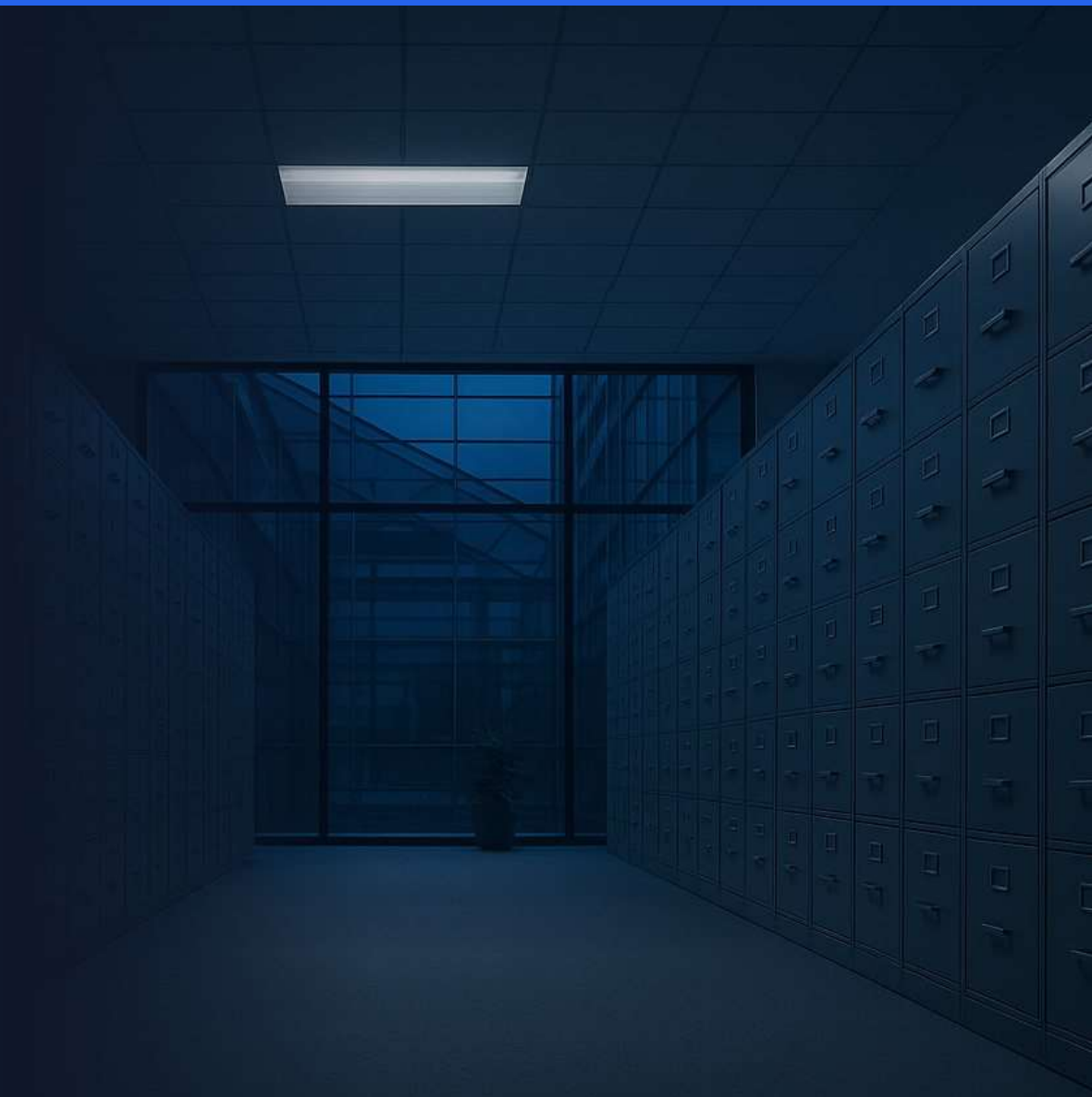
Route Models Across Workloads

How to send each request to the right model, balancing quality, speed, cost and the consequences of being wrong.

Theory session

Beginner

Running case: LumaForge Robotics



ORIENTATION

Where this session sits

1.1 architecture, 1.2 data, and now the decision that runs on every request

MODULE 1
AI System Foundations

MODULE 2
Controlled AI Operations

MODULE 3
Strategic AI Operating Choices

SESSIONS IN THIS MODULE

1.1 Architect a Multi-Model Product
How specialised models cooperate inside one product

1.2 Establish Synthetic Data Trust
Traceable data, and knowing which evidence is real

1.3 **Route Models Across Workloads**
Choosing a model by task, quality, latency and cost

YOU ARE HERE

SESSION OUTCOMES

What this session explains

Choosing which model handles each piece of work

01

Routing is a decision you make thousands of times a day

Small differences in the rule produce large differences in cost and quality.

02

Four inputs decide the route

How hard the task is, how good the answer must be, how fast, and what it costs to be wrong.

03

Cascading is the default pattern

Try cheap first; escalate only what the cheap model is not confident about.

04

A routing policy must be measurable

If you cannot see what each route costs and achieves, you cannot improve it.

DEFINITION

What a routing policy is

A rule, written down, that can be inspected and changed

WORKING DEFINITION

A routing policy is the rule that decides, for each incoming request, which model or path handles it — based on the task type, the quality required, the time available, and the cost of an error.

1

It is explicit

Written down and version-controlled, not implicit in whichever model someone wired in first.

2

It is measured

Each route reports its volume, cost, latency and quality separately.

3

It is changeable

Changing the policy should not require changing the models — Session 1.1's loose coupling.

CONCEPT 1 · THE TRADE-OFF

Quality, speed and cost — pick your point

No route wins on all three, so the policy decides which to give up

QUALITY	How often the answer is good enough for the task at hand.
LATENCY	How long the requester waits. Sometimes a hard limit, not a preference.
COST	What each request consumes. Multiplied by volume, this becomes the business model.
RISK	What it costs when the answer is wrong. The dimension most often left out.

THE BEGINNER'S MISTAKE

Optimising quality everywhere

Using the strongest model for every request is simple, and usually makes the product unaffordable.

THE BETTER QUESTION

What is good enough here?

Most requests do not need the best possible answer. A few need much more than the default.

THE MISSING DIMENSION

Cost of being wrong

A wrong shelf label is an annoyance. A wrong grasp plan is a damaged product or an injury.

CONCEPT 2 · THE MAP

Two questions place almost every request

How hard is it, and how much does an error cost?

COST OF BEING WRONG ▲

VERIFY Easy but consequential. Cheap model plus a check, or a rule.	**BEST MODEL, PLUS REVIEW** Hard and consequential. Strongest model, human escalation — Session 2.2.
CHEAPEST ROUTE Easy and low stakes. The small model, or a cached answer.	**STRONG MODEL** Hard but recoverable. Pay for capability; accept occasional errors.

DIFFICULTY OF THE TASK →

WHY THIS WORKS

Volume sits bottom-left

In most products the majority of requests are easy and low stakes — which is where the savings are.

WHAT TO BUILD FIRST

The top-right rule

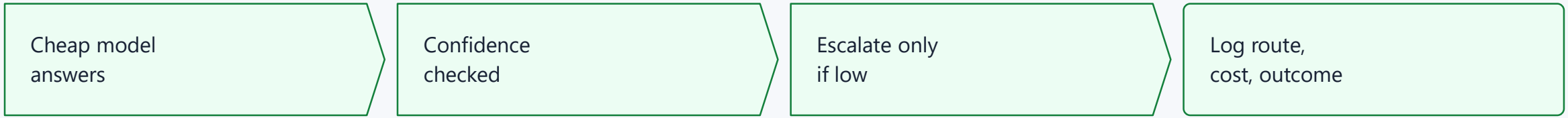
Define the high-stakes route before the cheap one. Getting that wrong is what costs you.

CONCEPT 3 · CASCADING

Try cheap first, escalate when unsure

The most useful routing pattern, and the easiest to start with

THE CASCADE



THE CONDITION THAT MAKES IT SAFE

This only works if the cheap model can say when it is unsure. A model that is confidently wrong makes a cascade worse than no cascade at all — which is why Session 1.1 insisted every output carries a confidence.

Chow, C. (1970). On optimum recognition error and reject tradeoff. IEEE Transactions on Information Theory, 16(1), 41–46.

CONCEPT 4 · SIGNALS

What the router can actually look at

Five signals, from cheapest to most informative

Signal	What it tells you	What it costs to get
Request type	Which category of work this is.	Nothing — it is already in the request.
Input size or complexity	A rough proxy for how hard the task will be.	Nothing; a simple measurement.
Customer or context tier	How consequential this particular request is.	Nothing, if your system records who is asking.
A small classifier	A learned estimate of difficulty, better than a rule.	One cheap inference per request.
The cheap model's own confidence	The most informative signal available.	A full cheap inference — which you often wanted anyway.

THE ORDER TO ADD THEM IN

Start with the free signals. Add a classifier only when you can show the free ones are leaving real money or real quality on the table.

CONCEPT 5 · EVIDENCE

Improving a policy that already works

FM Logistic's gains came on top of a mature baseline

REPORTED OUTCOMES AT FM LOGISTIC

Achieved against a baseline the company already considered highly optimised, and moved from pilot into production.

THE OLD POLICY

One step at a time, on cost

A greedy rule: take the cheapest next move. Functional, and blind to the whole route.

THE METHOD

Search, not tuning

Candidate algorithms were generated and refined automatically, rather than the existing rule being adjusted.

THE LESSON

“Already optimised” is a claim to test

A mature policy is evidence that something works, not evidence that nothing better exists.

CONCEPT 6 · AGGRESSIVENESS

How hard to push cheap routes

The dial most teams set once and never revisit

TOO CONSERVATIVE

Almost everything goes to the strongest model. Quality is fine; the unit economics are not.

TUNED

The cheap route handles the volume; escalation catches what it should. Reviewed monthly.

TOO AGGRESSIVE

Cheap routes handle work they cannot do. Savings show up immediately; the quality cost arrives later.

HOW TO FIND THE SETTING

Practice	What it gives you
Shadow the expensive model on a sample.	A direct measure of what the cheap route is getting wrong.
Set a quality floor, then minimise cost under it.	Makes the trade-off explicit instead of accidental.
Review the threshold on a schedule.	Models, prices and traffic all change; a fixed threshold silently drifts out of tune.

CONCEPT 7 · FAILURE MODES

Three routing mistakes that are hard to see

Each shows up in the numbers long after it starts

Routing on cost alone

Looks like A rule that always picks the cheapest model that fits.

Why it fails It ignores what an error costs, which is the whole reason some requests are different.

Fix Put stakes in the policy explicitly, as a separate input.

No per-route measurement

Looks like One overall quality number for the product.

Why it fails A failing route hides inside a healthy average.

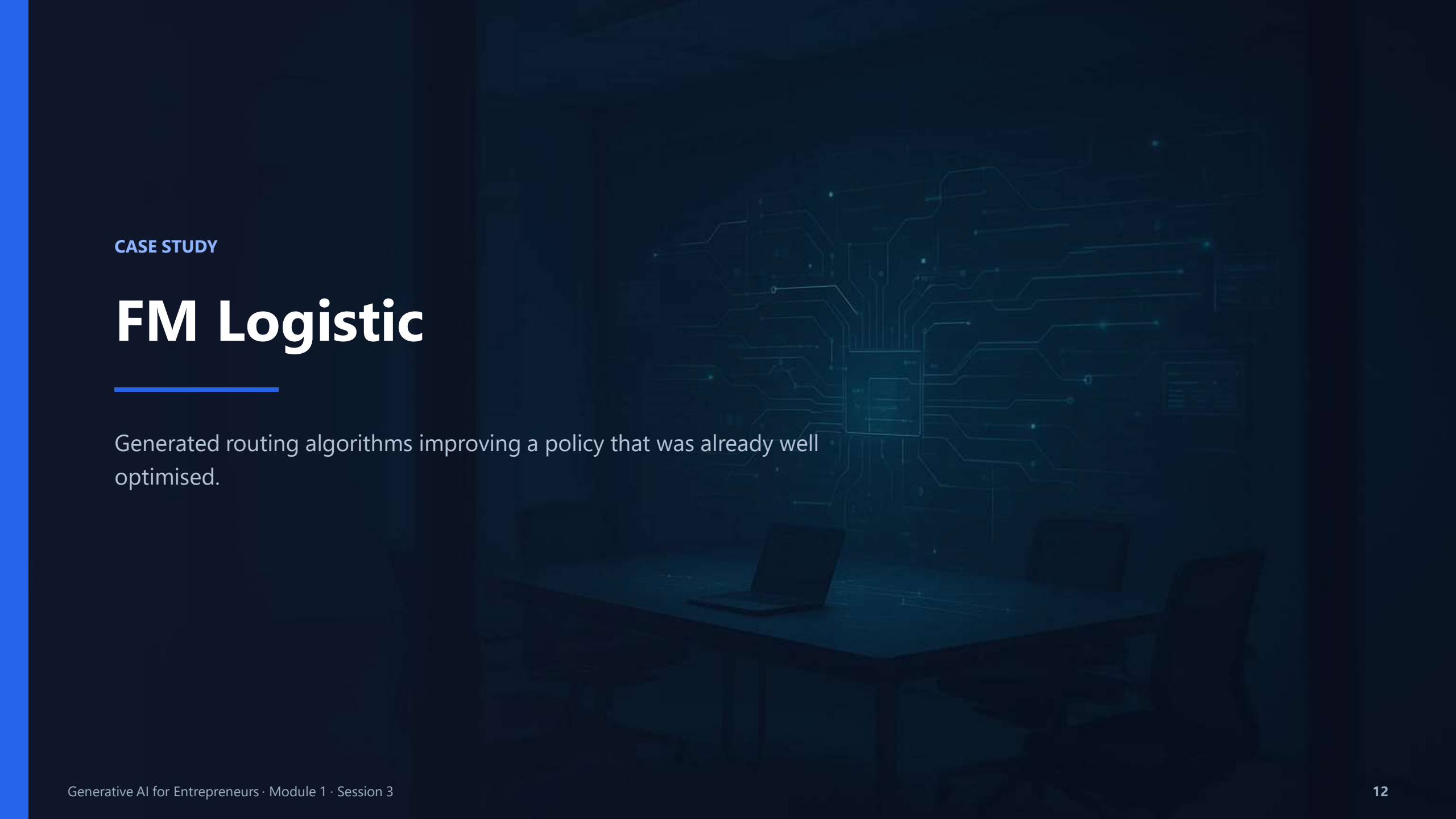
Fix Report volume, cost, latency and quality for each route separately.

A threshold set once

Looks like A confidence cut-off chosen at launch.

Why it fails Traffic, prices and model behaviour all change around it.

Fix Re-derive it on a schedule from current shadow-test data.



CASE STUDY

FM Logistic

Generated routing algorithms improving a policy that was already well optimised.

CASE STUDY · FM LOGISTIC

10.4% better routing on an optimised baseline

Searching for a better rule instead of tuning the existing one

THE STARTING POINT

A functioning, optimised baseline

THE BUSINESS

Logistics and warehouse operations, where routing logic shapes how fulfilment work moves through a site.

THE OLD POLICY

Decisions made one step at a time, prioritising cost. Functional, and mature.

THE CONSTRAINT

Improve it without treating the existing approach as disposable.

THE RESULT

Better logic, found by search

THE METHOD

AlphaEvolve on Google Cloud autonomously generated and refined candidate algorithms.

THE GAIN

A 10.4% improvement in routing efficiency, on top of an already highly optimised baseline.

AT SCALE

Over 15,000 kilometres less travel per year; the pilot moved into production.

READING THE CASE

A working policy can still leave real room for improvement. The gain came from searching the space of routing rules rather than tuning the existing one.

Drawn from the session case pack.

APPLICATION

Bringing the theory to LumaForge Robotics

Writing the routing policy for the warehouse assistant

Step	What you decide	LumaForge example
List the request types	The distinct kinds of work arriving.	Shelf reading, grasp planning, operator questions, exception review.
Place each on the map	Difficulty against cost of error.	Operator questions: easy, low stakes. Grasp planning: hard, high stakes.
Assign a default route	Cheapest model that meets the quality floor.	Operator questions go to the small model with a cached answer layer.
Define escalation	The confidence level at which work moves up.	Any grasp plan below the confidence threshold goes to the strong model, then a human.
Instrument every route	Volume, cost, latency, quality — separately.	A weekly table with one row per route and its four numbers.
Shadow-test monthly	Run the expensive model on a sample of cheap-route traffic.	One per cent of shelf readings double-run, and the threshold re-derived.

WHAT COMES NEXT

Module 2 takes this operating system and asks how you know it is still working — and what it may decide alone.

SUMMARY

What to carry out of this session

1. **Routing balances four things, not one.**
   Quality, latency, cost and the cost of being wrong.

2. **Most requests do not need the best model.**
   The volume sits in the easy, low-stakes corner — which is where the savings are.

3. **Cascade: cheap first, escalate when unsure.**
   It only works if the cheap model can report low confidence.

4. **Measure each route separately.**
   A failing route hides inside a healthy overall average.

5. **“Already optimised” is a claim worth testing.**
   FM Logistic found 10.4% on a baseline it considered mature.

FURTHER READING

Sources for this session

Primary references behind each concept

REJECT OPTION Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1), 41–46.

LEARNING TO DEFER Madras, D., Pitassi, T. & Zemel, R. (2018). Predict responsibly: improving fairness and accuracy by learning to defer. *NeurIPS 31*.

CASCADES Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR 2001*.

COST-AWARE SERVING Chen, L., Zaharia, M. & Zou, J. (2023). FrugalGPT: how to use large language models while reducing cost and improving performance. *arXiv:2305.05176*.

CALIBRATION Guo, C. et al. (2017). On calibration of modern neural networks. *ICML 2017*.

OPERATING PRACTICE Beyer, B. et al. (2016). *Site Reliability Engineering*. Sebastopol: O'Reilly.

ML SYSTEMS Huyen, C. (2022). *Designing Machine Learning Systems*. Sebastopol: O'Reilly.