Finding the 80/20

...

Lessons from delivering our first LLM feature

Thordis Thorsteins
Lead AI Data Scientist

Agenda

1. The experience
2. The prioritisation
3. What worked well and what didn’t
4. Takeaways

Thordis Thorsteins

Pre-launch Readiness Checklist for LLM Features

1. Code & Architecture

- [ ] All feature code merged
- [ ] Prompt(s) reviewed and committed to Git

Feature flags created for:

  - [ ] Feature enablement/disablement
  - [ ] Model selection
  - [ ] Prompt(s)

- [ ] Performance tracking and alerts configured
- [ ] Usage tracking configured

2. Security & Privacy

- [ ] OWASP security review completed
- [ ] Pentest completed
- [ ] New vendors approved

3. Evaluation & Quality

- [ ] Offline evaluations passed
- [ ] SME qualitative evaluation passed

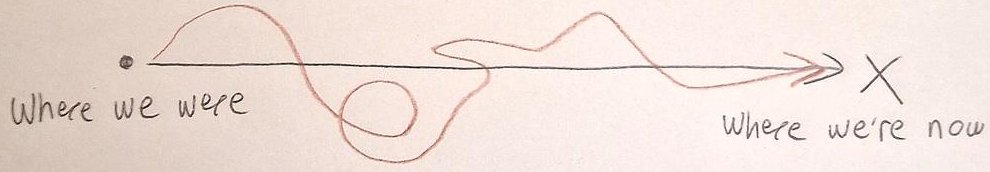
4. Customer Collateral

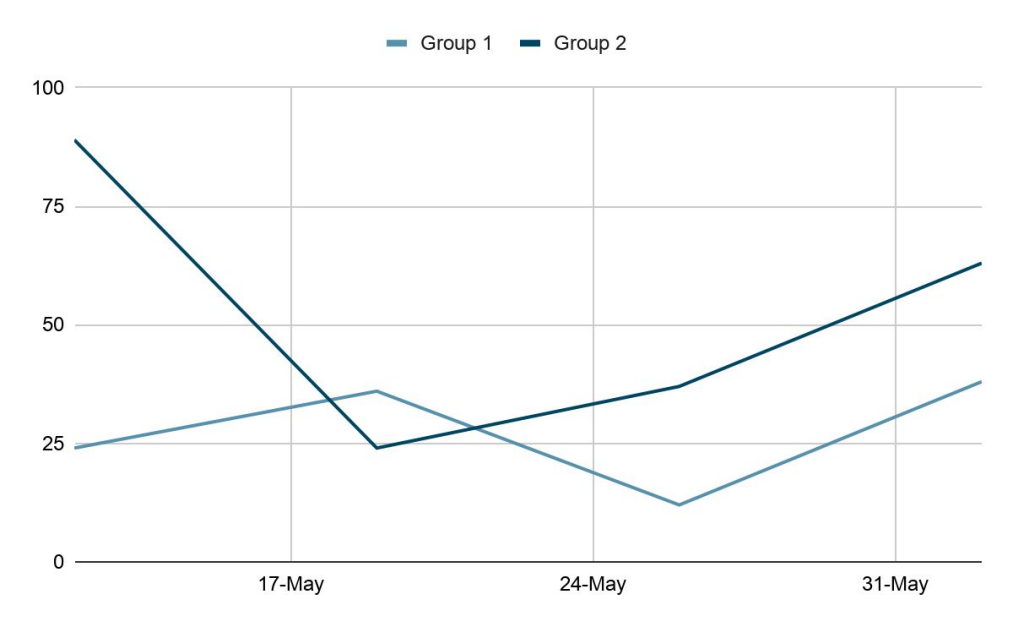
- [ ] AI disclaimer added to UI
- [ ] FAQ created and added to Help Center
- [ ] Announcement email sent to customers 2 weeks prior to enablement

5. Support & Internal Enablement

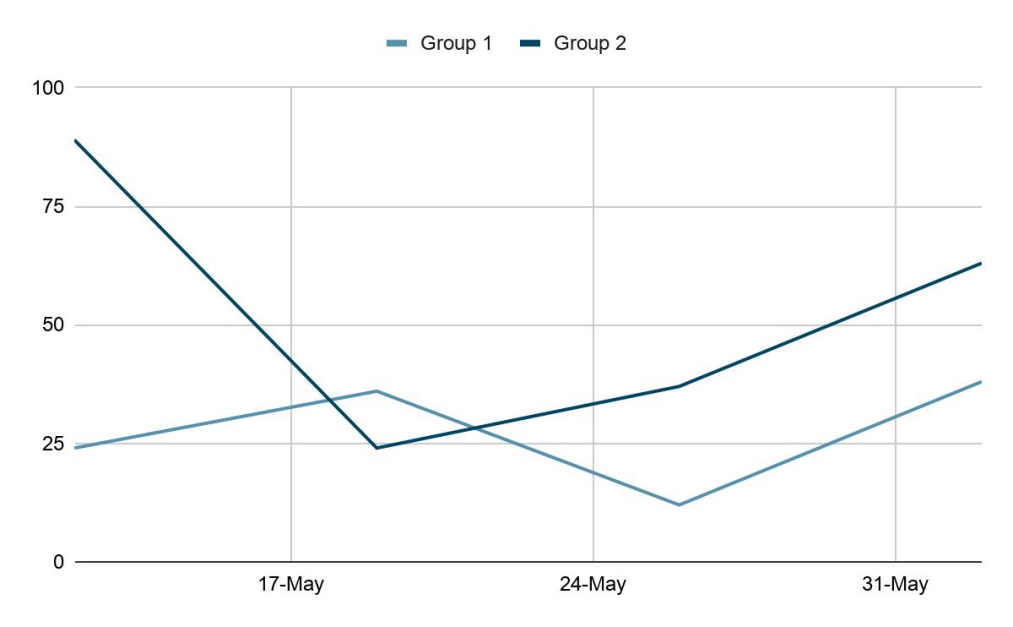
- [ ] Everything here all at once page updated
- [ ] Internal enablement session held for support teams


Where we were
Where we're now


Where we were
Where we're now



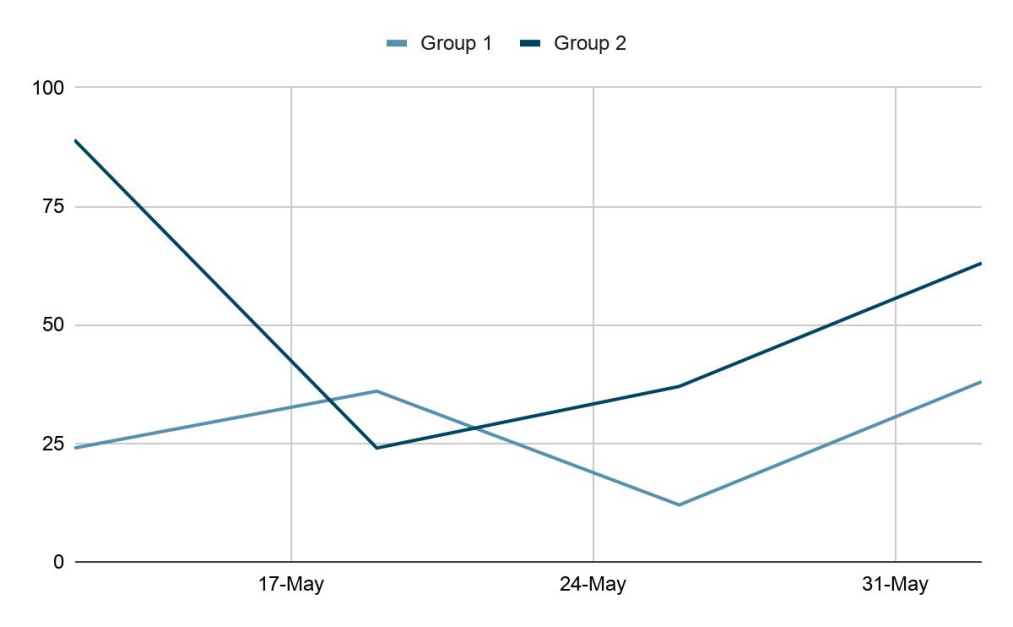
Thordis Thorsteins



[Explanation of trend]

[Recommended actions]

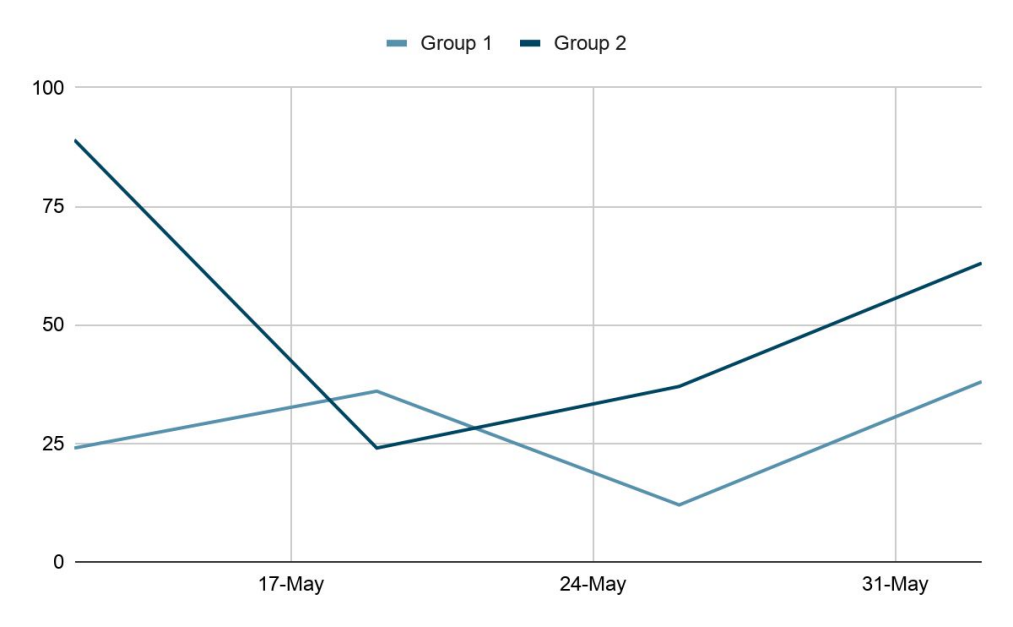
Thordis Thorsteins



[**Better** explanation of trend]

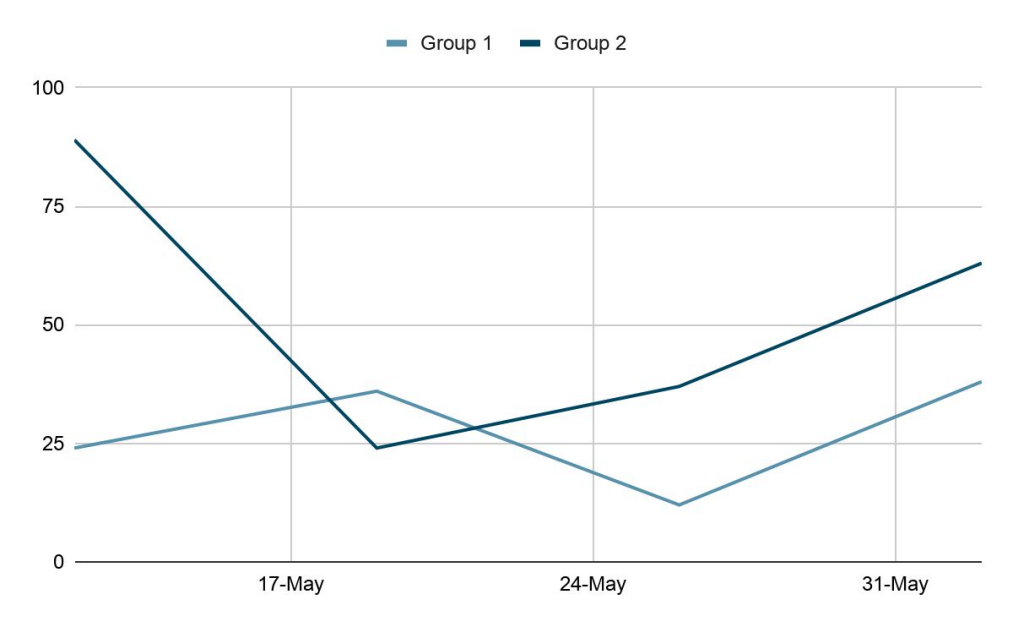
[Recommended actions]

Thordis Thorsteins



[**Even** better explanation of trend]

[Recommended actions]

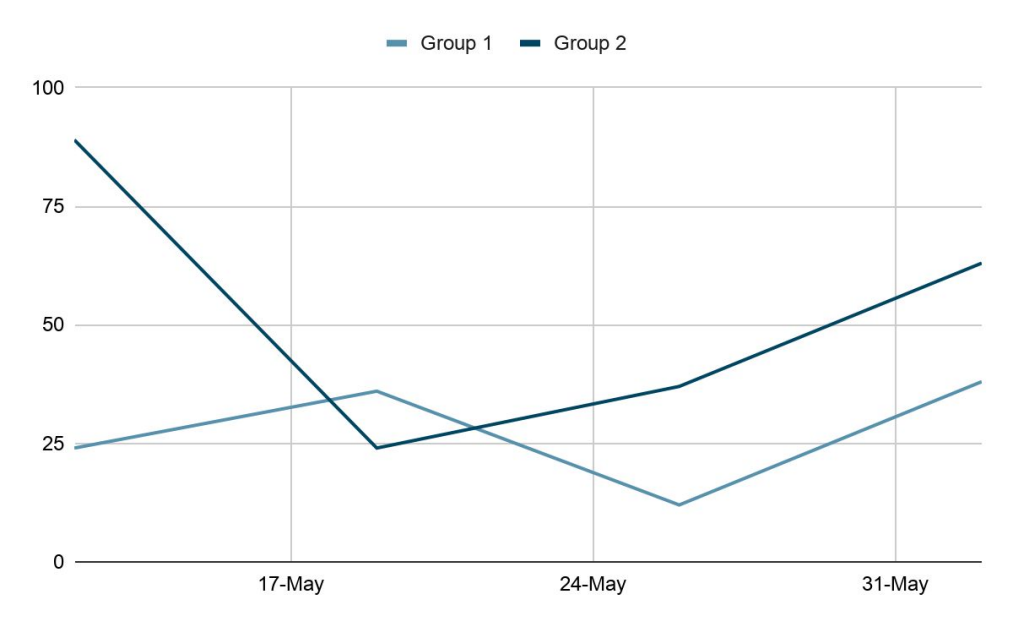


Problems

When is the commentary “good enough”?

[Even better explanation of trend]

[Recommended actions]



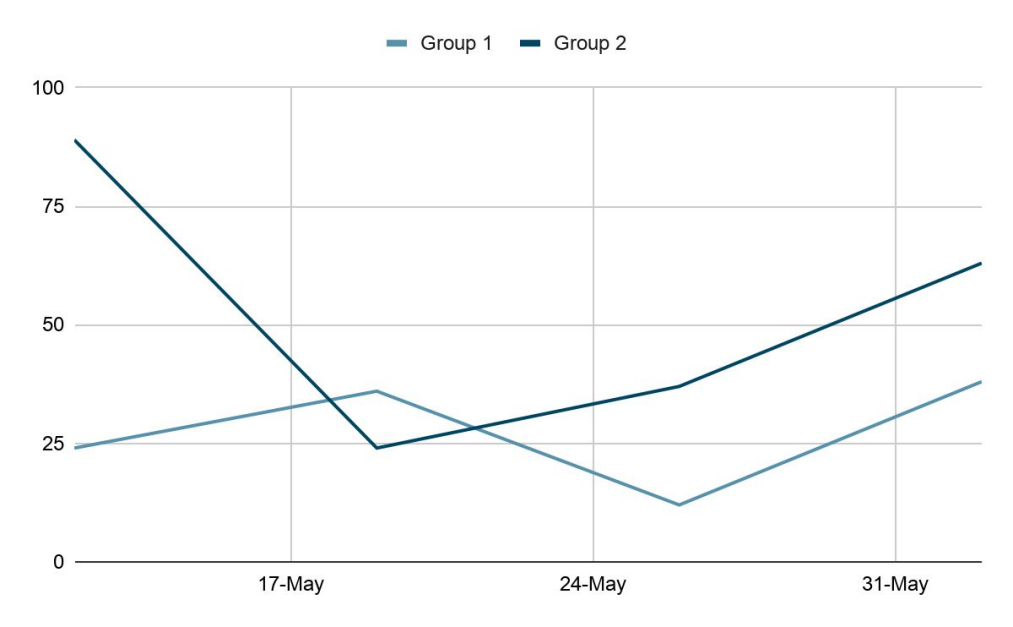
Problems

When is the commentary “good enough”?

Fix one thing, break another

[Even better explanation of trend]

[Recommended actions]



[Even better explanation of trend]

[Recommended actions]

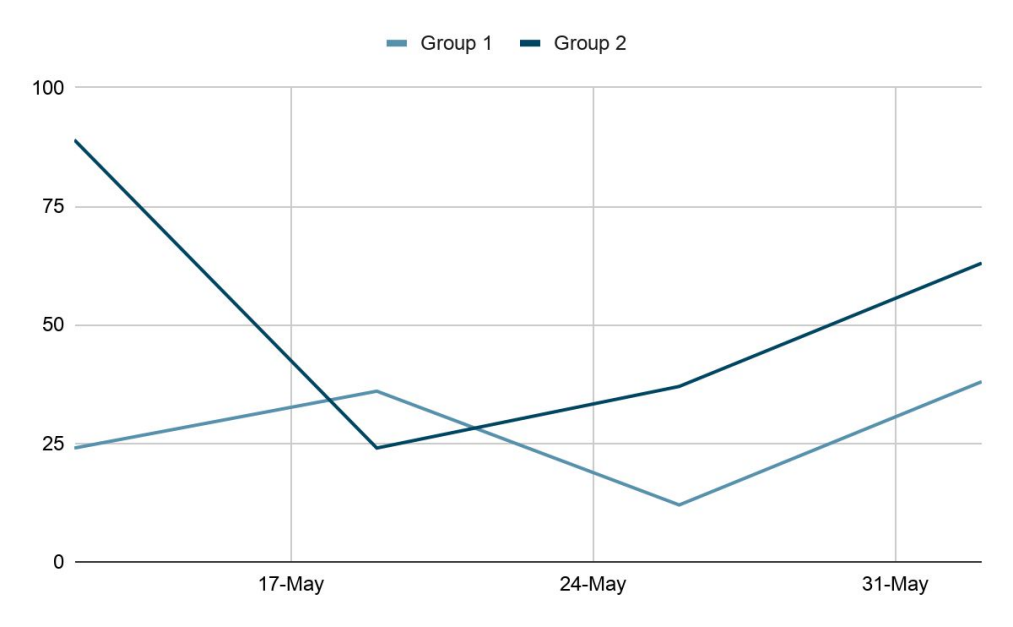
Problems

When is the commentary “good enough”?

Fix one thing, break another

Collaborative prompt iterations are... challenging

Thordis Thorsteins



[Even better explanation of trend]

[Recommended actions]

Dev Problems

When is the commentary “good enough”?

Fix one thing, break another

Collaborative prompt iterations are... challenging

Thordis Thorsteins

Non-dev Problems

Yesterday the commentary explained a dip. Today it doesn't

Evolving customer AI adoption processes

Conflict between earning trust and sharing too much

Dev Problems

When is the commentary "good enough"?

Fix one thing, break another

Collaborative prompt iterations are... challenging

Prompt conventions	Evaluation framework	Security review	Built-in vs. opt-in vs. opt-out stance
Prompt governance	Evaluation set-up	Legal review	External AI governance collateral
Context conventions	Observability design	Company AI stance	Customer upskilling support
Development lifecycle updates (fit for AI dev)	Infrastructure optimisation	Commercial team upskilling	Deployment process update
Dev team upskilling	Cost optimisation	Pricing	Etc. etc. etc.


Perfect feature

Feature that's good enough for our context
AND
easy to iterate on

Thordis Thorsteins


Prioritise
V1 now
Park

What's most important based on:

- Our product type and identity?
- Our customers' needs?
- Our teams' needs?

What's most important based on:

- Our product type and identity?
- Our customers' needs?
- Our teams' needs?

→ Security and trust

What's most important based on:

- Our product type and identity?
- Our customers' needs?
- Our teams' needs?

→ Security and trust

~~Speed, impressiveness, cost, breadth of functionality, ...~~

Security

Trust

Security

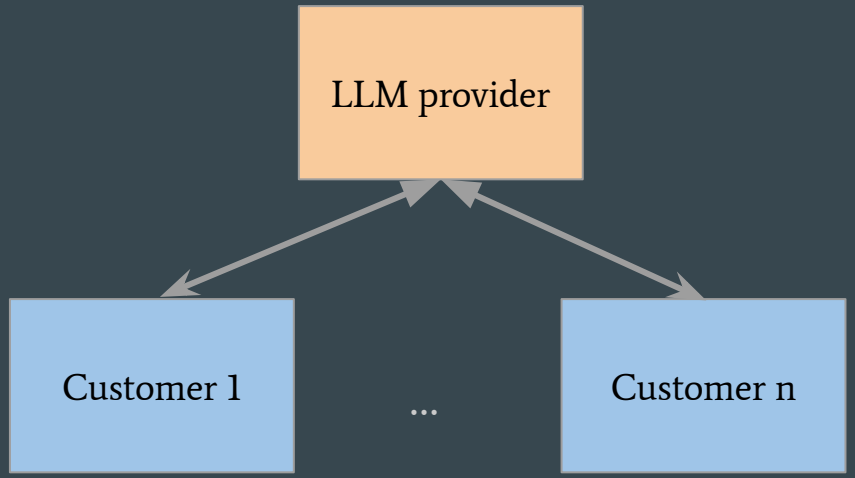
1. Data segregation architecture

Trust

Security

Trust

1. Data segregation architecture



Security

1. Data segregation architecture

Trust

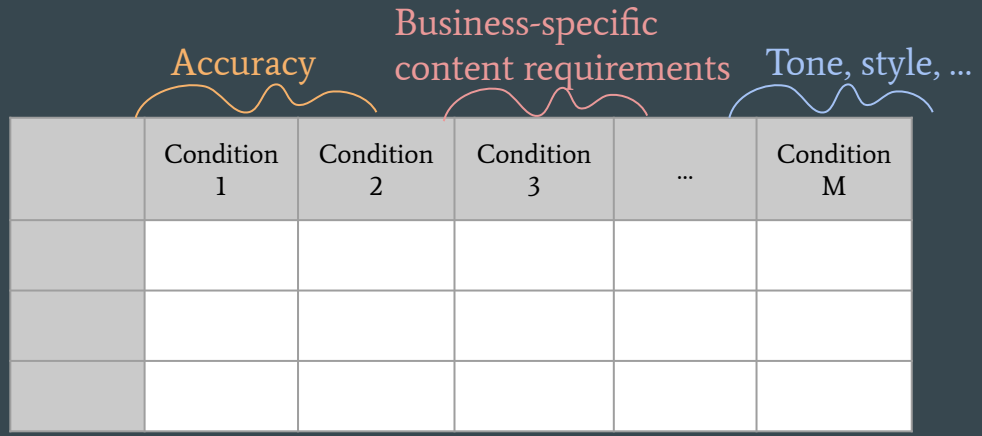
1. Human evaluation of outputs

Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs



Thordis Thorsteins

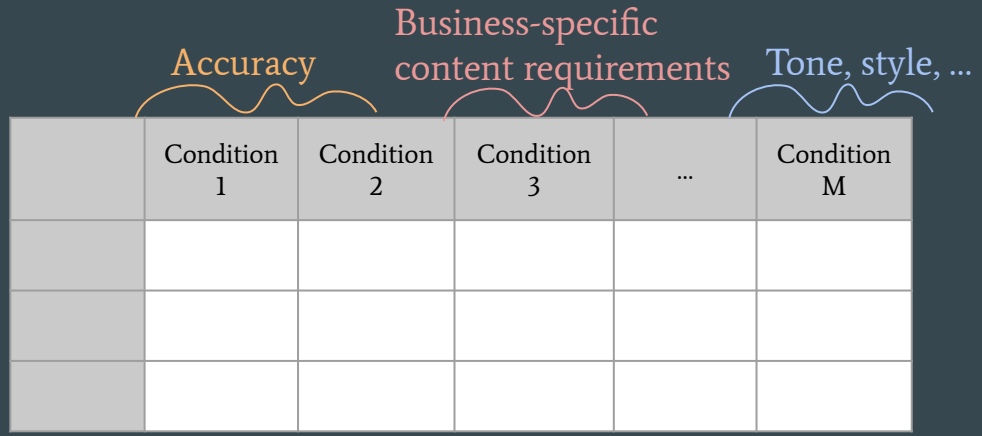
Security

1. Data segregation architecture

Input 1
Input 2
...
Input N

Trust

1. Human evaluation of outputs

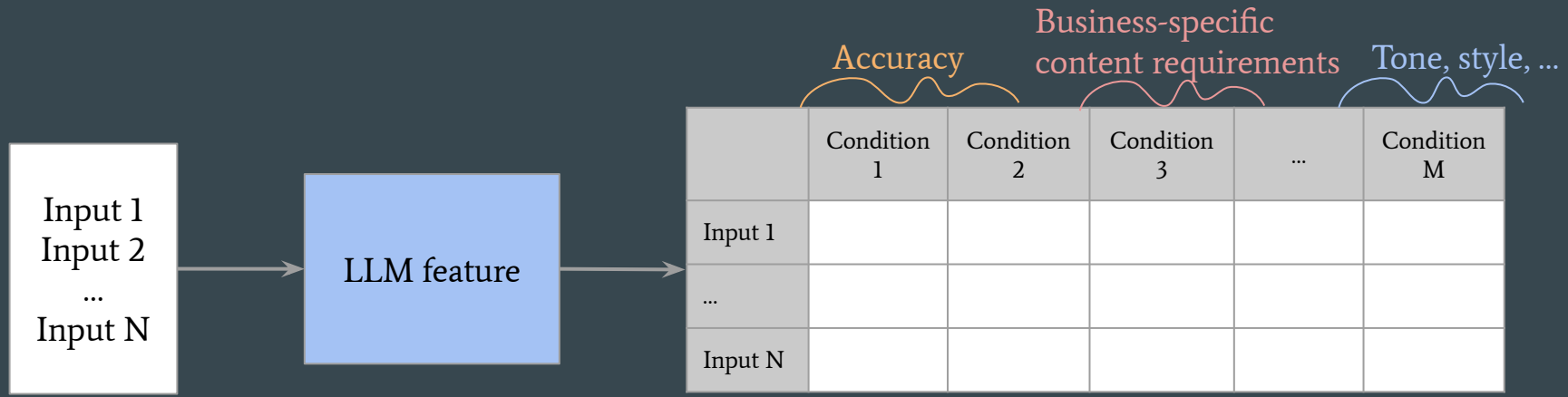


Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs

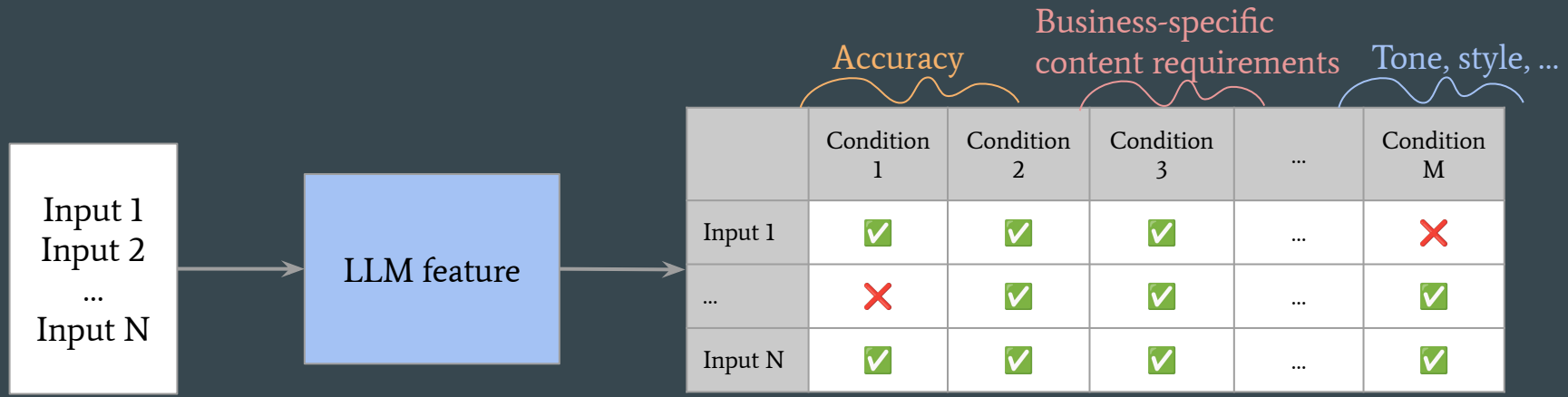


Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs



Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs
2. Deployment process definition

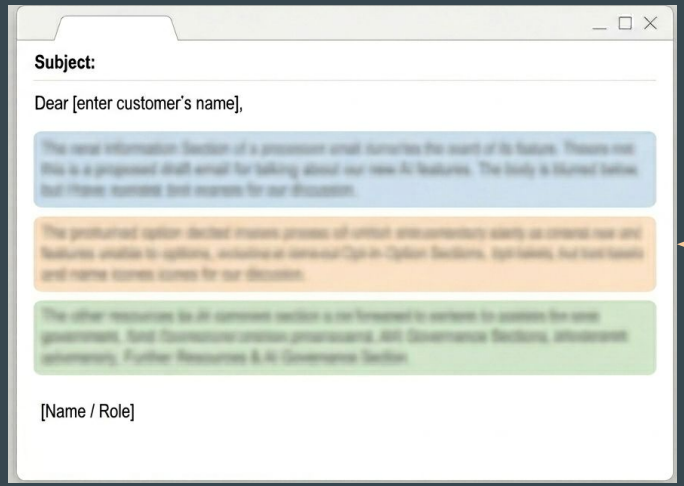
Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs
2. Deployment process definition

AI governance info



Built-in vs. opt-out vs. opt-in?

Security

1. Data segregation architecture

Trust

1. Human evaluation of outputs
2. Deployment process definition
3. AI FAQs

Security

1. Data segregation architecture
2. Security review against OWASP

Trust

1. Human evaluation of outputs
2. Deployment process definition
3. AI FAQs

Security

1. Data segregation architecture
2. Security review against OWASP

Trust

1. Human evaluation of outputs
2. Deployment process definition
3. AI FAQs

+ To make our future lives easier:
    - Feature flags
    - Basic usage tracking

Things we didn't prioritise

- ✗ Infrastructure
- ✗ Context optimisation
- ✗ Evaluation and observability frameworks
- ✗ Edge case coverage
- ✗ Cost optimisation and pricing
- ✗ Documentation for internal teams

Would do again

Wish we'd done sooner

Would do again

Wish we'd done sooner

- Internal documentation about:
  - what the feature does
  - how to talk about it
- Document decisions

Would do again

- Start with human evaluations
- Write the AI FAQ early
- Design for iteration
- Regular check-ins with collaborating teams

Wish we'd done sooner

- Internal documentation about:
  - what the feature does
  - how to talk about it
- Document decisions

Takeaways: Finding the 80/20 in a rapidly evolving problem space

Takeaways: Finding the 80/20 in a rapidly evolving problem space

1. Let go of the idea that you'll get it all right the first time

Takeaways: Finding the 80/20 in a rapidly evolving problem space

1. Let go of the idea that you'll get it all right the first time
2. Design for iteration

Takeaways: Finding the 80/20 in a rapidly evolving problem space

1. Let go of the idea that you'll get it all right the first time
2. Design for iteration
3. Work with a cross-functional team to agree where it's not acceptable to fail

Takeaways: Finding the 80/20 in a rapidly evolving problem space

1. Let go of the idea that you'll get it all right the first time
2. Design for iteration
3. Work with a cross-functional team to agree where it's not acceptable to fail
4. Prioritise making it difficult to fail in those areas

Takeaways: Finding the 80/20 in a rapidly evolving problem space

1. Let go of the idea that you'll get it all right the first time
2. Design for iteration
3. Work with a cross-functional team to agree where it's not acceptable to fail
4. Prioritise making it difficult to fail in those areas
5. Communicate decisions, changes and progress often

Thank you!