Furtherline

The product works. Make room for what comes next.

A practical guide for founders whose customers are starting to outgrow the technology.



Protect what works. Understand the limit. Choose the next step.

Furtherline

Start with the customer journey worth protecting.

Pick one journey that customers already value. Signing in, placing an order, uploading a file or completing a booking is a better starting point than “scale everything”.

What does the customer need to finish?

What would happen to the business if that journey stopped working?

Which parts already work well and must survive a change?

Write down the journey, the customer’s expected result and the part you want to preserve.

Furtherline

Make the slowdown specific.

“The product is slow” is a symptom. Separate a slow response from a busy period, a growing queue or work that never completes.

Which action gets slower, and when?

Does it affect everyone or a particular customer or account?

What changed before the problem appeared?

Can you reproduce it without using a customer’s private information?

Bring the affected action, the pattern and the business consequence. An engineer can then investigate where the time goes.

Furtherline

Make interruptions easier to understand.

A dependable product needs more than an absence of error messages. Your team needs to notice a failure, understand its effect and recover without guessing.

How do you find out that a customer is stuck?

Who can restore the service, and what do they need?

What happens when an outside service becomes unavailable?

Has the recovery path actually been tried?

Choose the interruption that matters most. Describe the customer’s experience and how your team handles it today.

Furtherline

Keep the progress in an AI-built product.

Your customers do not care how the first version was built. They care that it works. The next stage is understanding the system well enough to change it safely.

Can one customer ever see or change another customer's information?

What information would be lost if the application stopped today?

Which outside services does the product rely on?

How do you check a change before customers receive it?

Who knows how to restore the previous working version?

Treat an unanswered question as something to investigate. It is not, by itself, a reason to start over.

Furtherline

Choose the smallest change that removes the real limit.

A rewrite is one option. It should have a reason stronger than discomfort with the existing code.

Keep: does the current system already meet the next need?

Extend: can a focused addition remove the limit?

Simplify: is avoidable complexity making the product harder to run?

Replace: is there evidence that the limit cannot reasonably be resolved in place?

Compare the customer benefit, disruption and way to check the result for each plausible option. Include leaving the system alone.

Furtherline

Bring a brief that makes the next conversation useful.

You do not need a technical diagnosis before speaking to Furtherline. Bring the business context and the customer experience.

The product: who uses it and what it helps them do.

Demand: what people pay for and what brings them back.

The constraint: what gets slow, breaks or becomes difficult to change.

The effect: what this costs customers or your team.

The next stage: what you want to make possible.

Use the brief builder at furtherline.com/scaling-brief, or send a few sentences at furtherline.com/contact. We will start with the situation.