MODULE 2 · SESSION 1

Choose the Smallest Viable Product

How to decide what goes in, what waits, and in what order — so each build earns its cost in learning.

Theory session

Beginner

Running case: BrewMetric

ORIENTATION

Where this session sits

Module 1 validated the product; Module 2 decides how much of it to ship, and what it costs

MODULE 1
Product Build and Validation

MODULE 2
MVP Economics and Measurement

MODULE 3
Sales Launch and Operations

SESSIONS IN THIS MODULE

2.1 **Choose the Smallest Viable Product** — YOU ARE HERE
Deciding what goes in, what waits, and in what order

2.2 Set Prices That Protect Cash
Connecting price and packaging to margin, acquisition and runway

2.3 Instrument the Product for Learning
Measuring activation, retention and conversion after release

SESSION OUTCOMES

What this session explains

Sizing a first release so it teaches without exhausting you

01

Minimum and viable are both constraints

Dropping either word produces a different, worse product.

02

Scope follows uncertainty

Build only as much as the current open question requires.

03

Slice vertically

A narrow complete journey beats a broad unfinished one.

04

Exclusions must be recorded

What is deliberately out is as much a decision as what is in.

DEFINITION

What “minimum viable” actually means

Both halves of the phrase are doing work

WORKING DEFINITION

A minimum viable product is the smallest release that delivers real value to a real customer while producing the evidence you need for the next decision.

1

Minimum

Smallest relative to a stated question. Without the question, “minimum” has no meaning and becomes an argument about taste.

2

Viable

It must actually work for the customer. A broken product shipped early tests your tolerance for complaints, not your hypothesis.

3

Product

Someone outside the team uses it unsupervised. That is what separates it from the prototype in Session 1.2.

Ries, E. (2011). The Lean Startup. · Blank, S. & Dorf, B. (2012). The Startup Owner's Manual.

CONCEPT 1 · THE MISREADING

Minimum viable is not minimum quality

The most common failure is cutting the wrong dimension

WHAT YOU CUT

Scope

WHAT THIS MEANS

Fewer use cases, fewer segments, fewer configurations, fewer edge cases handled.

EFFECT ON THE CUSTOMER

The product does less — but what it does, it does properly.

EFFECT ON LEARNING

A clean signal: they either valued that narrow thing or they did not.

WHAT YOU MUST NOT CUT

Quality of what remains

WHAT THIS MEANS

Reliability, correctness, safety, and the basic dignity of the experience.

EFFECT ON THE CUSTOMER

They cannot tell whether the idea is bad or the build is — so they leave.

EFFECT ON LEARNING

The signal is uninterpretable, which makes the whole exercise worthless.

THE RULE

Cut the number of things the product does. Do not cut how well it does them. A narrow product that works teaches you something; a broad one that does not teaches you nothing.

CONCEPT 2 · SEQUENCING

Investment follows the open question

FastCafé's two builds are a worked example of this ladder

DOES ANYONE WANT THIS?

Cheapest possible mechanism. FastCafé used SMS — around 20 orders in a week.

DOES IT WORK IN CONTEXT?

Real setting, real users. Four cafés over three days.

WILL THEY PAY FOR IT?

Real money changing hands. About $700 in orders captured.

WILL THEY KEEP USING IT?

Repeat behaviour over time — the question Session 2.3 instruments for.

THE DISCIPLINE

One rung at a time

Building for rung three while rung one is unanswered is the most expensive mistake available here.

WHAT CHANGES EACH TIME

The question, not the ambition

FastCafé did not lower its sights. It raised investment only when the next question required it.

NAMING IT

Write the learning job

Every build gets one sentence: “this exists to find out whether…”.

CONCEPT 3 · FORMS

Four MVPs that need almost no product

Each substitutes something cheaper for the thing you would eventually build

1

Concierge

You deliver the service by hand, openly, to one customer at a time. Slow and unscalable — and it reveals the real steps the product must automate.

2

Wizard of Oz

The customer believes it is automated; a person is behind the curtain. Tests whether the output is valued before the machinery exists.

3

Single-feature

One capability, done properly, for one segment. FastCafé's SMS ordering is this: nothing but the core mechanism.

4

Piecemeal

Assembled from tools that already exist — forms, spreadsheets, messaging. Nothing is built; the service is real.

5

Pre-order or waitlist

Demand is measured before supply exists. Strongest when it collects a genuine cost from the customer.

CHOOSING BETWEEN THEM

Match the substitution

Ask what the expensive part is. Substitute exactly that, and build the rest honestly.

Typology after Ries, E. (2011) and Bland, D. & Osterwalder, A. (2019), Testing Business Ideas.

CONCEPT 4 · SLICING

Vertical slices, not horizontal layers

How you divide the work decides whether anything is usable at the end of it

HORIZONTAL

Build by layer

THE APPROACH

Finish the database, then the services, then the interface.

WHY IT IS TEMPTING

It matches how teams are organised and how work feels efficient.

THE PROBLEM

Nothing is usable until the last layer lands, so no feedback arrives until it is too late to act on.

VERTICAL

Build by journey

THE APPROACH

One complete path from trigger to value, through every layer.

WHY IT IS HARDER

It requires several skills at once and looks less efficient on a plan.

THE PAYOFF

Something real works early, so the riskiest assumption gets tested while there is still time.

THE WORKING DEFINITION

A vertical slice is the smallest thing you can put in front of a customer and honestly ask them to use.

Patton, J. (2014). User Story Mapping. Sebastopol: O'Reilly.

CONCEPT 5 · THE DECISION

In, next, or never

Three buckets, and the third one is what keeps the first honest

IN THIS RELEASE

Required for the vertical slice to deliver value and produce the signal you named.

EXPLICITLY NEXT

Wanted, understood, and deliberately deferred — with the condition that would pull it forward.

EXPLICITLY NEVER

Decided against, with the reason recorded so it is not relitigated every month.

WHAT EACH SCOPE DECISION MUST RECORD

Field	Why it is needed
The learning job	One sentence: what this release exists to find out.
The signal	The result that would change the next decision — stated before the build starts.
What is deferred, and why	An unrecorded exclusion returns as a surprise halfway through.
The date it stops	A release with no end date absorbs every good idea that arrives.

CONCEPT 6 · SCOPE CREEP

Why scope grows quietly

Each addition is individually reasonable, which is exactly the problem

Beyond the flattening point, each feature adds build time and delay without adding to what the release can tell you.

THE MECHANISM

No single culprit

Every addition is defensible on its own. The cost only appears in aggregate, by which point the date has moved.

THE DEFENCE

A fixed date

When the date is fixed, scope becomes the variable — and trade-offs get discussed rather than absorbed.

THE QUESTION TO ASK

“Which signal does this change?”

If a proposed feature changes no signal, it belongs in the next release.

CONCEPT 7 · VIABILITY

What “viable” has to mean for the customer

Four conditions that a narrow product still has to satisfy

Condition	The question	If it fails
Complete for one job	Can they finish a whole task, not part of one?	They abandon midway and you learn nothing about value.
Reliable	Does it work every time, or only in the demo path?	Failures get attributed to the idea rather than the build.
Understandable	Can they work it out without you present?	You measure your onboarding skill, not the product.
Honest	Are limitations stated up front?	Complaints crowd out the signal you were trying to collect.

WHY ALL FOUR

A narrow product that satisfies all four teaches you more than a broad one that satisfies three.

CONCEPT 8 · FAILURE MODES

Three MVPs that waste the release

All three ship. None of them produce a usable answer.

The small version of everything

Looks like Every planned feature, each half-built.

Why it fails No journey completes, so no value is delivered.

Fix One vertical slice, finished.

The unmeasured launch

Looks like A real product, shipped, with no instrumentation.

Why it fails You have customers and no evidence.

Fix Decide the signal before the build — Session 2.3.

The permanent MVP

Looks like A release that was never meant to last, still running two years later.

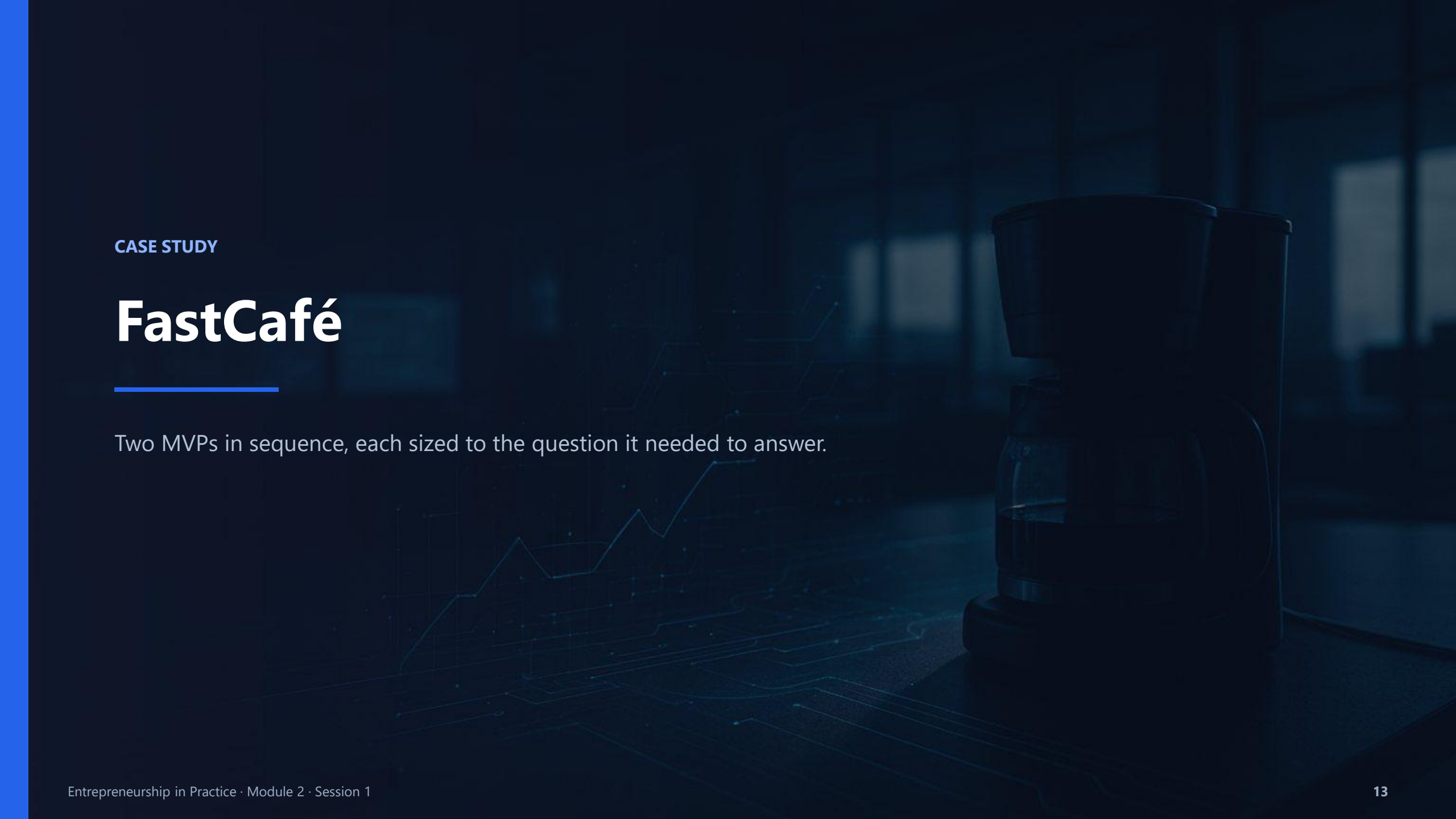
Why it fails Temporary shortcuts become the architecture.

Fix Set a date to decide: extend, rebuild, or stop.

CASE STUDY

FastCafé

Two MVPs in sequence, each sized to the question it needed to answer.



CASE STUDY · FASTCAFÉ

Two builds, two questions

Investment rose only when the question required it

FIRST BUILD

SMS ordering

THE QUESTION

Would people use a simple ordering mechanism at all?

WHAT WAS BUILT

An SMS ordering system, tested internally with company staff. No app, no interface.

THE RESULT

Around 20 orders in one week — an observation, not a proof that a coffee service was ready.

SECOND BUILD

HTML mobile app

THE NEW QUESTION

How does the concept behave in real cafés, with real choices?

WHAT WAS BUILT

An HTML mobile app letting users pick coffee, café and pick-up time; tested at 4 cafés over 3 days.

THE RESULT

Approximately $700 in orders captured — a richer signal, earned by a richer build.

READING THE CASE

The interface was never the achievement. Each build had a stated learning job, and investment rose only when the question demanded it.

Drawn from the session case pack.

APPLICATION

Bringing the theory to BrewMetric

Sizing a first release around one open question

Decision	What you choose	BrewMetric example
The learning job	One sentence, written first.	Find out whether café staff will log a brew if it takes under ten seconds.
The form	The cheapest substitution that still answers it.	Piecemeal: a tablet form and a shared sheet. No connected hardware.
The slice	One complete journey, end to end.	From starting a brew to seeing this week's consistency summary.
Explicitly deferred	Recorded, with the condition that would pull it forward.	Automatic capture — deferred until manual logging proves the summary is valued.
The signal	Stated before the build.	At least half of shifts logging on four of five days, for two weeks.
The stop date	Fixed, so scope is the variable.	Three weeks from start, whatever is finished.

WHAT COMES NEXT

Session 2.2 asks what this release should cost and what it should charge.

SUMMARY

What to carry out of this session

1. **Minimum is meaningless without a question.**
   Smallest relative to what you are trying to find out — otherwise it is just an argument about taste.

2. **Cut scope, never quality.**
   A broken product makes the result uninterpretable, which wastes the whole release.

3. **Slice vertically.**
   One complete journey teaches you more than every feature half-built.

4. **Raise investment one rung at a time.**
   FastCafé built the app only after SMS had answered the first question.

5. **Write down what you are not building.**
   An unrecorded exclusion comes back as a surprise mid-build.

FURTHER READING

Sources for this session

Primary references behind each concept

MVP	Ries, E. (2011). *The Lean Startup.* New York: Crown Business.
CUSTOMER DEVELOPMENT	Blank, S. & Dorf, B. (2012). *The Startup Owner's Manual.* K&S Ranch.
SLICING	Patton, J. (2014). *User Story Mapping.* Sebastopol: O'Reilly.
EXPERIMENTS	Bland, D. & Osterwalder, A. (2019). *Testing Business Ideas.* Hoboken: Wiley.
SCOPE CONTROL	Beck, K. (2000). *Extreme Programming Explained.* Boston: Addison-Wesley.
PRIORITISATION	Cohn, M. (2005). *Agile Estimating and Planning.* Upper Saddle River: Prentice Hall.
CRITIQUE	Eisenmann, T., Ries, E. & Dillard, S. (2012). *Hypothesis-Driven Entrepreneurship.* Harvard Business School Note 812-095.