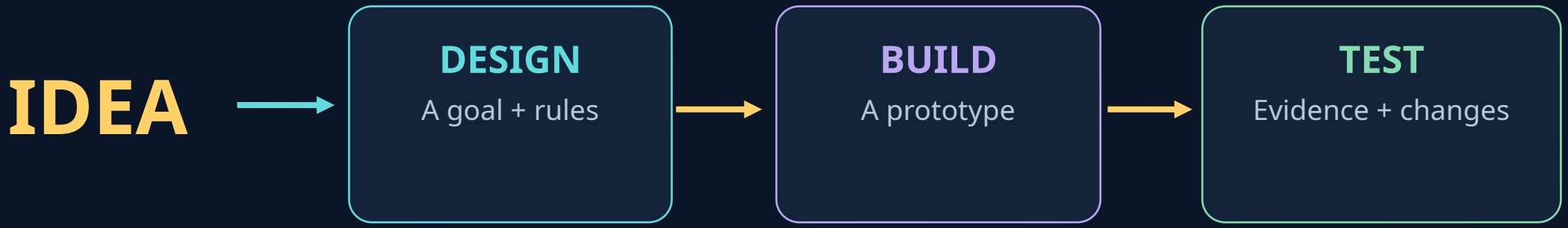
How Does an App or Video Game Become Real?

From a human idea to a working, testable experience — even with just paper.



No programming required to complete this lesson. Bring your curiosity.

Choose your depth, not your rank

Anyone can follow along alone, with a friend, in the room, or on a replay.

EXPLORER

Understand how an idea becomes a screen, an action, and a result. Use everyday language.

BUILDER

Make three screens, write rules, then let someone test your paper prototype.

ENGINEER / RESEARCHER

Define states, edge cases, invariants, performance needs, and evidence of correctness.

SUCCESS = an answer, an artifact, a next step, or a better question

Seven weeks led us here

You already know more of the app-building story than you might think.

01 W1: A technology experience is a system

02 W2: One tap becomes an event and response

03 W3: Speed and reliability change how it feels

04 W4: Hardware and operating systems do the work

05 W5: Networks connect local and remote components

06 W6: AI is one possible tool inside an experience

07 W7: Security protects people and information

A useful app feels simple. Building it takes decisions.

Imagine an app that helps a community member find the next technology workshop.

WHAT THE PERSON SEES

“Find Thursday’s session.”

One clear button.
A useful answer.
A way to ask for help.

WHAT THE TEAM DESIGNS

Audience and purpose.
Information and rules.
Screens, inputs, data.
Accessibility, privacy, tests.

The making process is a loop, not a straight line

An idea becomes stronger when real users try it and you learn from them.



Return to an earlier step whenever evidence shows something needs to change.

Apps and games share the same building blocks

Their goals can differ, but both translate human actions into software behavior.

APP EXAMPLE

A library finder helps someone locate a book, check availability, or request help.

Success: the person completes a useful task.

GAME EXAMPLE

A tiny “Signal Sprint” game asks a player to collect stars while avoiding clouds.

Success: the player understands the rules and can play.

Both need: inputs • rules • state • outputs • feedback • tests

Before code: Who is this for?

The best first question is not “Which programming language?”

PERSON

Who will use it? What experience, access, and devices might they have?

NEED

What problem are they trying to solve, or what experience do they want?

RESULT

How could we tell they successfully accomplished their goal?

Describe the experience in one sentence

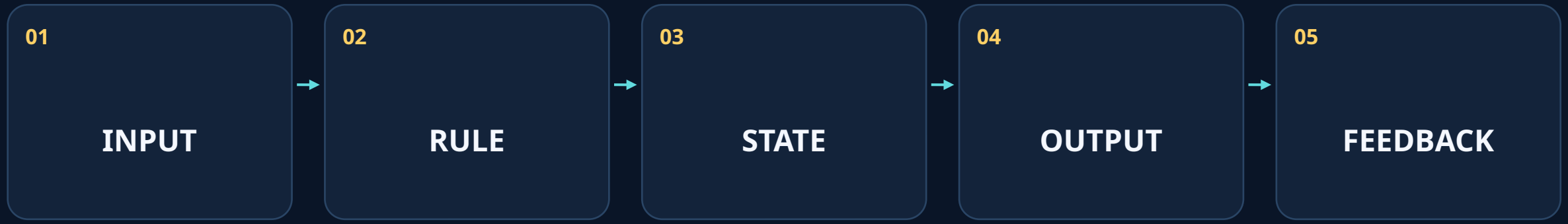
A user story is a starting conversation, not a complete specification.

AS A [person], I WANT TO [action], SO THAT [benefit].

Example: As a learner, I want to find a free workshop so I can attend confidently.

Five pieces make something interactive

These ideas exist in websites, apps, games, kiosks, and learning systems.



Example: TAP “PLAY” → apply rules → change playing status → show video → show response

State means what the experience remembers

A rule uses inputs and state to decide what happens next.

BEFORE

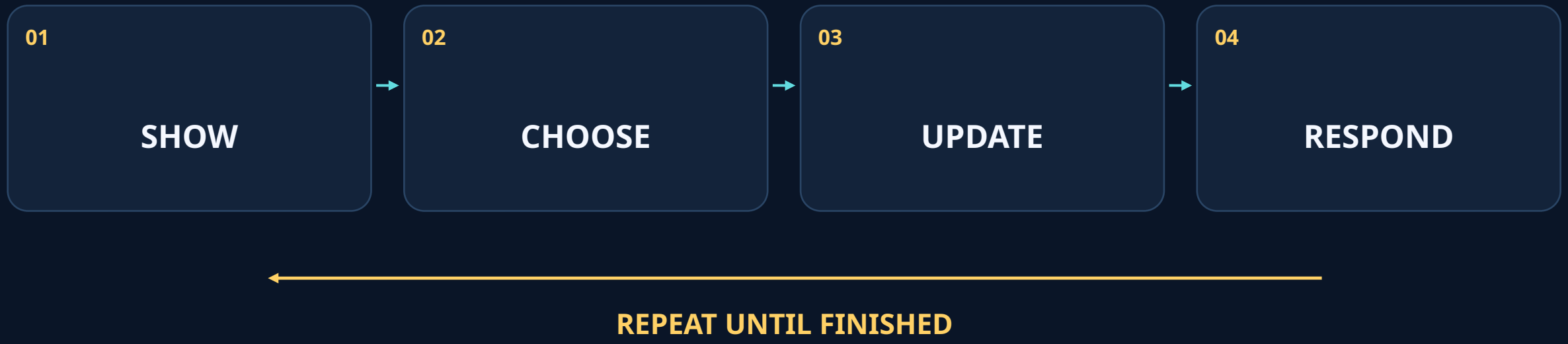
Score: 0
Round: 1
Card: STAR
Choice: not yet made

AFTER “COLLECT”

Score: 1
Round: 2
Card: CLOUD
Message: “+1 star!”

The person and the system take turns

Interactive experiences repeat this loop until a goal or stopping condition occurs.



Meet Signal Sprint: a game you can play on paper

Three turns. One decision each turn. Track your score without a computer.

GOAL

Finish three turns with a score of at least 2 stars.

CHOICES

On each card: COLLECT or SKIP.

CARDS

We use a fixed sequence: STAR → CLOUD → STAR. No dice or secret choices.

A fair game tells you exactly what choices do

One consistent rule set makes the example playable and testable.

01 Start with score = 0 and round = 1.

02 STAR + COLLECT → score increases by 1.

03 CLOUD + COLLECT → score decreases by 1.

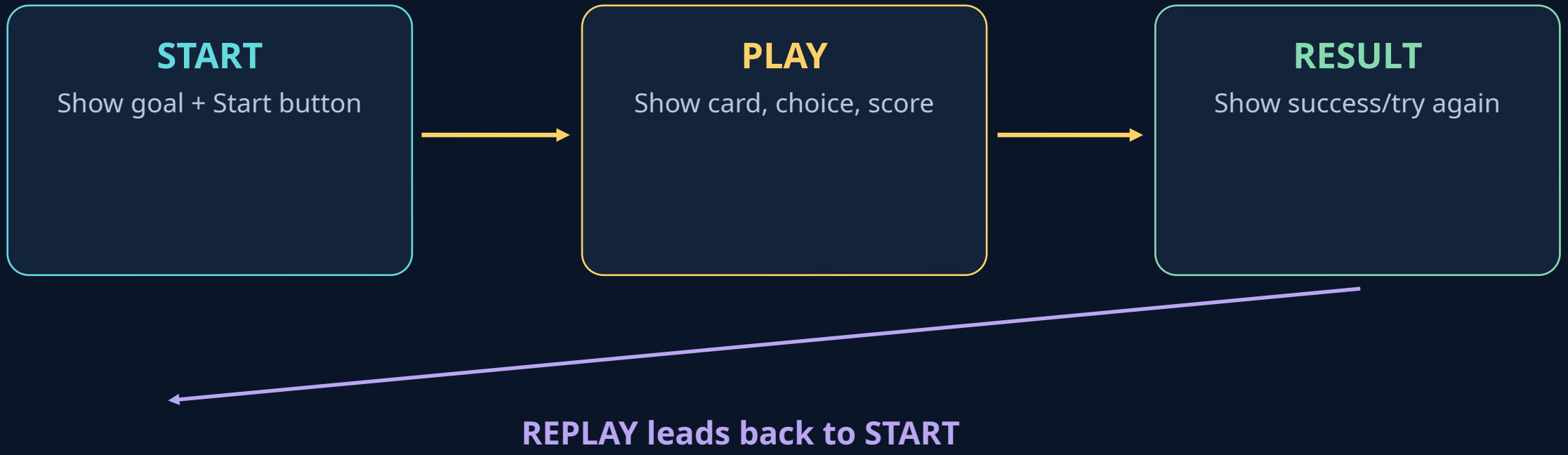
04 SKIP → score stays the same (for either card).

05 After each choice, move to the next card.

06 After three rounds, score ≥ 2 means SUCCESS; otherwise TRY AGAIN.

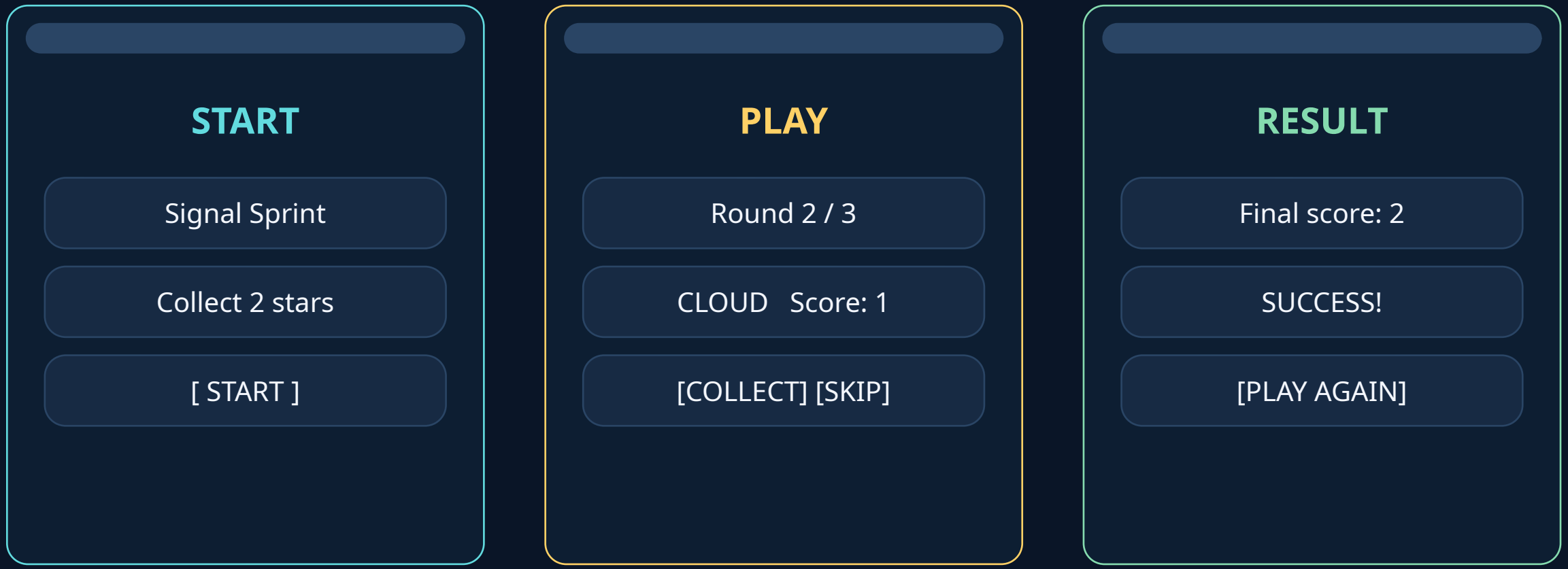
Three screens can hold an entire small experience

A state diagram shows which screen or mode comes next.



Paper screens are real design tools

A prototype can reveal confusing choices before any code exists.



Rules become simple decisions

You can write the logic in everyday language before learning a programming language.

```
WHEN a player chooses an action:
  IF action is COLLECT and card is STAR: score = score + 1
  IF action is COLLECT and card is CLOUD: score = score - 1
  IF action is SKIP: score stays the same
  THEN move to the next round
  AFTER round 3: show success if score >= 2
```

Art, sound, writing, and accessibility matter

A product feels usable because many kinds of decisions work together.

EXPERIENCE

Clear text, understandable buttons, meaningful feedback, helpful instructions.

CREATIVE WORK

Characters, visuals, sounds, motion, narrative, and cohesive style.

INCLUSION

Readable contrast, captions, alternatives to sound, keyboard/touch paths, no color-only meaning.

“Did it work?” is a design question

Every action should make the result understandable, including errors and delays.

UNCLEAR

You tap COLLECT.
Nothing changes.
You tap again.
Was the first tap recorded?

CLEAR

You tap COLLECT.
The score updates.
“+1 STAR” appears.
The next card is shown.

A paper prototype proves an idea is understandable

It does NOT prove a game or app is ready to ship.

PROTOTYPE CAN TEST

Is the goal clear?
Can people make choices?
Are the rules coherent?
Can we observe confusion?

SHIPPING ALSO NEEDS

Working implementation.
Accessibility and privacy.
Robust failure handling.
Device and platform testing.
Ongoing maintenance.

A test compares expected and observed behavior

“It seems fine” is not evidence that a system works correctly.

ACTION

What did the person do?
Example: Collect a STAR.

EXPECTED

What do our rules predict?
Example: 0 → 1 point.

OBSERVED

What actually happened?
Example: score changed to 1.

Can we win by following the rules?

Worked example using the public STAR → CLOUD → STAR sequence.

ROUND	CARD	CHOICE	SCORE AFTER
Start	—	—	0
1	STAR	COLLECT	1
2	CLOUD	SKIP	1
3	**STAR**	**COLLECT**	**2**

Final score = 2 → SUCCESS

What if the player does the unexpected?

A robust design must decide how to respond at its boundaries.

DOUBLE TAP

Should two rapid taps on COLLECT count once or twice? Decide and test.

NO SELECTION

What happens if someone waits or closes the game? Show an understandable state.

LOW SCORE

A score can become negative. Is that intended? If not, change the rule and test it.

A bug is a mismatch worth investigating

If reality and the intended rule disagree, write down what you saw.

EXPECTED: Score stays 1 after SKIP on a CLOUD.

OBSERVED: Score becomes 0.

Investigate the state update; do not rewrite the rule to hide the defect.

Test with another person without coaching them

Watch what they actually do, not what you hoped they would do.

GIVE THE GOAL

“Try to win Signal Sprint.” Do not explain every button immediately.

OBSERVE

Notice hesitation, wrong turns, repeated taps, and unexpected interpretations.

ASK AFTER

“What did you think would happen?” “What felt confusing?”

Improve ONE thing, then test again

Changing many things at once makes it hard to know what helped.

BEFORE

Some players miss the score.

Hypothesis: the score is hard to find.

AFTER

Move the score near the choice buttons and make it larger.

Measure: can more testers notice it?

One experience, more ways to participate

Accessibility benefits learners with different devices, abilities, and environments.

01 Write button labels; do not rely on color alone.

02 Ensure instructions can be read and understood.

03 Make targets large enough for touch or paper play.

04 Provide alternatives to audio-only clues and motion.

05 Allow time and clear progress feedback.

06 Test with people whose needs differ from yours.

Do not collect what you do not need

A helpful app or game can be designed with privacy and safety from the beginning.

GOOD DEFAULTS

Use fictional names in demos.
Ask only for needed information.
Protect saved work.
Make sharing understandable.

AVOID

Real passwords in class activities.
Unnecessary location tracking.
Publishing classmates' data.
Unexplained permissions.

A correct game can still feel wrong

Response time, frame timing, crashes, and connection quality shape enjoyment.

RESPONSIVENESS

How quickly does an action show meaningful feedback?

RELIABILITY

What happens if the app closes or an operation fails?

CONSISTENCY

Does the experience work similarly across supported devices?

Software creation includes many kinds of work

Different skills are needed even when one person takes multiple roles.

DESIGN + CONTENT

Understand needs, sketch interactions, write instructions, create art and sound.

ENGINEERING

Implement behavior, manage data, devices, performance, and connections.

QUALITY + OPERATIONS

Test, document, release, support, maintain, and improve.

How a real release reaches people

A public product needs evidence beyond one successful classroom test.



Store submission, rights, privacy, accessibility, platform rules, updates, and support require real work.

Invent an idea in three sentences

You have everything needed: paper, pencil, and imagination.

01 My user is _____

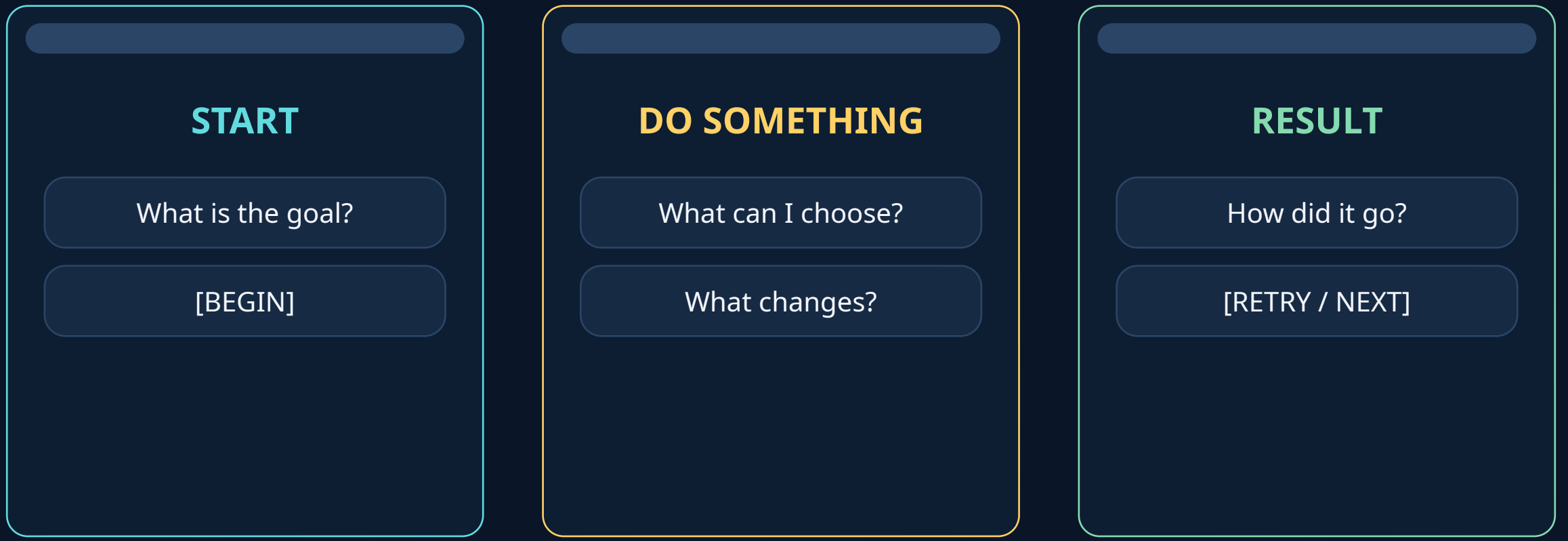
02 They want to _____

03 The experience succeeds when _____

Examples: a study timer • a community-event finder • a tiny collecting game

Draw three screens and one interaction

Your screens can be boxes with words; they do not need polished art.



Test it, and let the evidence speak

Choose a normal action, an unexpected action, and one improvement.

TEST 1	Normal action	Expected result? / Observed result?
TEST 2	Unexpected action	Expected result? / Observed result?
CHANGE	One improvement	What will change? / How will I retest?

You now have a design + prototype + test evidence.

One complete playthrough of Signal Sprint

A sample trace: STAR / COLLECT, CLOUD / SKIP, STAR / COLLECT.

01 Before play: score = 0, round = 1.

02 Round 1: COLLECT STAR → score 1.

03 Round 2: SKIP CLOUD → score remains 1.

04 Round 3: COLLECT STAR → score 2.

05 Three rounds finished; 2 ≥ 2 → SUCCESS.

Other sequences are possible; their scores follow the same published rules.

Engineering version: models, invariants, and failure paths

The same paper design can become a precise implementation plan.

STATE MACHINE

Modes: START, PLAY, RESULT.
Transitions happen on explicit actions.

INVARIANTS

Exactly one card per round.
Exactly three completed rounds.
Score updates once per valid choice.

FAILURE HANDLING

Reject duplicate actions.
Preserve or reset state deliberately.
Make errors visible.

From curiosity to creation

Your paper prototype can become a project, portfolio artifact, or group discussion.

KEEP THE ARTIFACT

Save your user story, three screens, rules, and test log. That is real design evidence.

KEEP EXPLORING

gunnchos.com/labs/thursday
WAIKE curriculum
The Technology Landscape

NEXT PUBLIC SESSION

DEC 3: Where Could I Fit in Technology?

No in-person Open Lab on Thanksgiving, Nov 26.

Bring a question. Bring a problem. Bring an idea. Or just bring your curiosity.

Keep exploring after the session

The public lesson and activity are free to revisit at your own pace.

gunnchOS3k + OPEN LAB

Program home, public lessons, and the broader technology ecosystem.

gunnchos.com/labs/thursday

WAIKE CURRICULUM

Go from this activity to structured technology learning pathways.

github.com/gunnchOS3k/waike-research-ops

THE TECHNOLOGY LANDSCAPE

Explore how people, systems, components, code, and networks fit together.

github.com/gunnchOS3k/gunnchos-technology-landscape