Jitto Full Stack Engineering Challenge: J250

Are Streaks Real?

Background

In this challenge, you'll build a full stack web app that explores a common idea people often assume is true:

> If someone succeeds multiple times in a row, are they more likely to succeed again?

We often think that when something happens more than once in a row, it means it will happen again, like a pattern has started. For example, if someone wins two games in a row, we might think they're "on a roll" and will win the next one too.

But is this actually true? Or does it just seem that way because of how we look at outcomes?

This app helps users investigate that question by generating long series of yes-or-no outcomes and looking at what happens after streaks occur. The system repeats this process thousands of times and shows how the results change as more data comes in.

The goal is for users to test this pattern for themselves using real data they generate through the tool.

The Process

Here's how it works:

1. The user chooses a success rate (for example, 0.3, 0.5, or 0.7).
2. The backend generates many sequences of 100 independent outcomes. Each outcome is either a success or a failure.
3. In each sequence:
   - Look for any case where two successes happen back to back.
   - Record what happened on the next outcome, if there is one.
4. Repeat this entire process across many sequences (for example, 10,000 times).
5. Track how the results change as more sequences are added.

The goal is to answer:

> After two successes in a row, how often does a third success happen right after?

At the end, the app compares this number to the original success rate the user selected. Are they close? Are they higher? Lower? How does it change over time?

Let the user explore and decide for themselves.

Your Task

Build a full stack web application with:

- A React frontend (written in TypeScript, styled with Tailwind or plain CSS)
- A serverless AWS backend (using Lambda, API Gateway, and CloudFormation)
- A backend function that generates random binary sequences and analyzes them
- A live display that shows how the answer changes as more data is processed

---

Frontend Requirements

Frameworks

- React
- TypeScript
- Tailwind CSS or plain CSS
- No external libraries

Must include

- A form that allows users to:
  - Choose a success rate between 0.1 and 0.9
  - Set how many sequences to run (for example, 10,000)
  - Enter a random seed (optional, to repeat results)
- A "Run Analysis" button
- A visual display of the results that shows:
  - How the estimated value changes as more sequences are added
  - The final estimate for "How often did a success follow two successes in a row?"
  - The difference between that number and the original success rate

You must design your own visualizations. You can use line graphs, number boxes, or step-by-step displays. Choose whatever makes the pattern and results easiest to understand.

Backend Requirements

Tools

- AWS Lambda (Python 3.12 with the AWS SDK for Python (Boto3), no external libraries)
- API Gateway (HTTP endpoint)
- CloudFormation (template must be exported from the AWS Console, not CDK or SAM)

Your function must:

- Accept a success rate (float between 0.1 and 0.9), a number of sequences, and an optional random seed
- For each sequence:
  - Generate a list of 100 random binary outcomes based on the success rate
  - Scan the sequence for any case where two successes happened back to back
  - For each such case, check the next outcome (if one exists)
  - Record whether that next outcome was a success or failure
- Return a list of result snapshots, each containing:
  - How many sequences have been processed
  - The current estimate of "success after two successes"
  - The original input success rate
  - The difference between the two

Important

- Do not count overlapping cases more than once.
- If the sequence ends after two successes and there is no next outcome, skip that case.

Output Format

You may design your own response structure, but it should include the following information for the frontend to use:

- Snapshots at regular intervals (for example, every 1,000 sequences)
- The final estimate and summary difference

Example:

```
{
  "snapshots": [
    {
      "sequences": 1000,
      "estimate": 0.49,
      "input_rate": 0.50,
      "difference": -0.01
    },
    {
```

```
      "sequences": 2000,
      "estimate": 0.502,
      "input_rate": 0.50,
      "difference": 0.002
    }
  ],
  "final": {
    "estimate": 0.503,
    "input_rate": 0.500,
    "difference": 0.003
  }
}
```

You can choose different key names or structure as long as the frontend can display the data clearly.

Bonus Features (Optional)

If you finish the core features, try one or more of the following.

Frontend

- Let users compare two runs side by side (for example, different success rates)
- Add a "Replay" button to step through results slowly
- Add keyboard shortcuts for moving forward or backward
- Make the interface work well on mobile

Backend

- Add a WebSocket API Gateway endpoint
  - Send snapshot updates in real time to the frontend
  - Use a simple format
- Add result caching
  - If a user runs the same configuration again, return a saved result instead of recalculating it

---

Discussion Questions

After you've built and tested your app, take some time to think about what you found. Include your answers in your README.

1. Before running the experiment, what did you expect to happen? If the success rate was 50%, how often did you think a third success would follow two in a row?
2. Did the results match your expectations? Were they higher, lower, or about the same? Why do you think that happened?
3. What changed when you used different success rates, like 30% or 70%? Did the pattern you saw get stronger, weaker, or disappear?
4. Can this kind of pattern be misunderstood in real life? Can you think of a real-world example where someone might notice a streak and assume something about what happens next?
5. Is the system changing, or is it just the way we are looking at it? Do you think the process "knows" it had two successes in a row? Or are we seeing something because of how we picked which outcomes to measure?

These questions do not have right or wrong answers. What matters is how you reason through them.

What to Submit

Push the following to a public GitHub repository:

1. All frontend code (React, TypeScript, Tailwind or plain CSS)

2. All backend code (Python 3.12, no external libraries)
3. A CloudFormation template (YAML or JSON) exported directly from the AWS Console
4. A README file that includes:
    - Setup instructions to run the app locally or deploy it
    - Notes on how you approached the architecture and edge cases
    - An estimate of your backend costs (API Gateway, Lambda) and how they grow with more simulations
    - Screenshots or logs from the AWS Console showing your live deployment (for example, Lambda logs or test responses)
    - Your answers to the discussion questions above
5. A live link to the deployed app (any public hosting platform is fine)

We are excited to see how you work with data, build on the web, and create something that is clear, correct, and fast from end to end. Good luck!