Coding Exercise - Vendor api router engine

At UrbanCompany marketplace platform, multiple teams are working in areas where they need to integrate with third party apis. In each such area the team integrates with multiple vendors which have different availability and costing.
One way to integrate for these needs is “if the primary vendor fails, try the next backup vendor to get the work done”. But this approach results in two things:

1. It increases overall API Response time for UC’s consumers (as all vendors are tried in sync one after another)
2. It increases UC’s cost

We need to build a smart router engine for such integrations. Which should try to get the response as much as possible from the first vendor only. That would mean the engine has to keep a track of how a particular vendor is performing and in case of higher failure it would switch to the next back-up vendor as primary till the last primary comes back fully.

For the first version, our product team is willing to give inputs to the system how to achieve it. Let’s start with it.
In a steady-state (all vendors are performing well), we want to distribute traffic in some ratio to be given by the team.
In case of any vendor facing high failures, we want to reduce primary traffic to that vendor to 10% and pass on the 90% of that vendor’s traffic to the next available vendor.
High failure threshold = Failed request % >= X in last Y minutes
Coming back threshold = Succeeded request % >= A in last B minutes

Let’s take an example.
Three vendors in order of their stack rank (vendor1, vendor2, vendor3)
For simplicity, the last ranked vendor would be always available (probably a very high cost vendor).
Steady-state traffic: vendor1 = 80%, vendor2 = 20%, vendor3 = 0%
X = 40%
Y = 5
A = 80%
B = 5

So in case in a particular duration only vendor1 is considered down by the definition of “High failure threshold”, new traffic would be
vendor1 = 8% (80 * 0.1), vendor2 = 92% (20 + 80 * 0.9), vendor3 = 0%

In case in a particular duration only vendor2 is considered down by the definition of “High failure threshold”, new traffic would be
vendor1 = 80%, vendor2 = 2% (20 * 0.1), vendor3 = 18% (20 * 0.9)

In case in a particular duration both vendor1 and vendor2 are considered down by the definition of “High failure threshold”, new traffic would be
vendor1 = 8% (80 * 0.1), vendor2 = 2% (20 * 0.1), vendor3 = 90% (80 * 0.9 + 20 * 0.9)

INPUT
Sample Data of 300 minutes duration
Vendors data - Time (Minutes), API Available
Data is given for every minute in the interval [0, 299] whether a particular vendor is up in that minute or not.
E.g.
Time (Minutes), API Available

0, True	->	Vendor api is up for first minute (i.e. for seconds [0, 59])
1, False	->	Vendor api is down for second minute (i.e. for seconds [60, 119])
7, True	->	Vendor api is up for eighth minute (i.e. for seconds [420, 479])

API Available = false for a particular minute means vendor api is down for the whole of that minute.
vendor1.csv
vendor2.csv
vendor3.csv

Request data - Request Index, Request Time (Seconds)
Data is given how requests are coming to UC in increasing order of time
A sample of requests coming to UC during 300 minutes interval [0 - 17999 secs]
Request Index, Request Time (Seconds)

0, 1	->	first request came at 1st second
1, 5	->	second request came at 5th second
2, 8	->	third request came at 8th second
3, 34	->	fourth request came at 34th second

request-time.csv
Consider [0-59] as the first minute, [60,119] as second minute and so and so on.

OUTPUT
For every request, all vendors tried (joined by |) in a csv file in following format:
Request Index, Vendors tried
0, vendor1
1, vendor1
2, vendor2
3, vendor1
.
.
.
.
76, vendor1|vendor2 (it means for 76th request, vendor1 api was down and it was served through vendor2)
.
.
147, vendor2|vendor3 (it means for 147th request, vendor2 api was down and it was served through vendor3)
.

.

Above example of output is just a sample, it’s independent of the input csvs provided.

Feel free to use whatever technologies you're the most comfortable with. This includes any sort of open-source third-party frameworks.
Your priorities should be:

- Workable solution (Shortcuts are fine, but be prepared to justify them and explain better solutions you would have implemented with more time)
- Clean Code & Design (How code is divided into proper readable separations)
- Extensibility of the solution (How easily can it solve for some other strategies to achieve similar / better outcome)

Also, plan to give a 45 minutes demo on your submission.
10 minutes - Showing your working solution in action and how it performs
15 minutes - Explaining approach and walking through the code
10 minutes - Talking about extensibility, other use cases or ways to solve this problem better
10 minutes - Q&A

Please reach out if you are stuck anywhere or need any help. We want this to be an enjoyable experience for you.