People's Prepaid — System Capability Report

lan89 · 11 scenarios · 2026-09-01 16:11 – 12:49

Generated 2026-11-25 11:51 · code version 9b177cf

1. What this report says

Each scenario below was built to answer exactly one question about the system, and is quoted only for that question. Together they describe how much the platform can sell, how many tills it can carry, and which component gives out first.

The campaign made 96,756 requests across 11 scenarios and sold 5,488 real vouchers. Every selling account was verified against the warehouse database as trading on the demo warehouse before any sale was made.

2. Capability — what the system can do

Question	Answer	From
Simultaneous READ-ONLY calls the transaction switch tolerates?	**held every level tested, worst p95 2,474 ms**	switch-probe, 2026-09-01 16:11
Peak selling throughput (tills selling flat out)?	**419.7 sales/min at 16 simultaneous sales**	voucher-sales-capacity, 2026-09-01 16:16
How many tills selling every 20 s?	**100 tills = 300 sales/min (~18,000/hour)**	voucher-sales-fleet, 2026-09-01 16:25
How many tills selling every 120 s?	**700 tills = 348.8 sales/min (~20,928/hour)**	voucher-sales-fleet-2min, 2026-09-01 16:33
How many tills can be CONNECTED (selling plus all background syncing)?	**800 tills at one sale per 120 s**	till-day-in-life, 2026-09-01 16:47
Raw request ceiling of the web tier?	**20 requests/s sustained, worst p95 3 ms**	health-baseline, 2026-10-05 12:42
Devices the settings store serves on their 20-second cycle?	**20 devices held, worst p95 7 ms**	fleet-poll, 2026-10-05 12:43
Ceiling for routine device chatter (ads, categories, update policy)?	**55 requests/s, worst p95 16 ms**	cached-feed, 2026-10-05 12:45
Database-writing device paths (impressions, update state)?	**10 requests/s, worst p95 14 ms**	device-secondary, 2026-10-05 12:46

3. Limits — where it stops, and what actually failed

Where	Limit	What failed	Attribution
Simultaneous selling	**32 tills**	95% of sales took up to 10813 ms (limit 8000 ms); a stalled sale held the step open 12s beyond the last answer	NOT the API — it peaked at only 25% CPU
Fleet at 20s cadence	**150 tills**	145 sales failed (40,4%, limit 5%); 95% of sales took up to 28705 ms (limit 8000 ms); the fleet got only 47,6% of the 450 sales/min it asked for (limit 90%); 123 of 359 till cycles (34,3%) overran their 20s slot (worst by 14,8s)	NOT the API — it peaked at only 27% CPU
Fleet at 120s cadence	**900 tills**	453 sales failed (94,8%, limit 5%); 95% of sales took up to 25297 ms (limit 8000 ms); the fleet got only 5,2% of the 450 sales/min it asked for (limit 90%)	NOT the API — it peaked at only 33% CPU

4. Where the load actually landed

Peak API CPU	Peak API memory	Peak proxy sockets	Peak test-rig CPU
49.4%	**354 MB**	**1,301**	**71.4%**

Across the whole campaign the API container never exceeded 49% CPU (worst during till-day-in-life). Note that Docker reports 100% as one core, so this is a fraction of the machine. The API is NOT the constraint anywhere in this report; the limits found are downstream of it — the transaction switch, a database, or a connection path. The load generator peaked at 71.4% CPU, so the figures

describe the server, not the test rig.

5. Reliability observations

Requests abandoned on timeout (switch-probe): 1 request(s) ran past the 30-second client limit and were given up on.

Recovery after overload (voucher-sales-capacity): Selling was responsive again 3 s after the overload stopped.

Stock reconciliation gap (voucher-sales-capacity): The warehouse recorded 1,307 vouchers leaving stock but the test received 1,302 — a difference of 5, i.e. sales the till gave up on that the switch completed anyway.

Requests abandoned on timeout (voucher-sales-capacity): 5 request(s) ran past the 30-second client limit and were given up on.

Recovery after overload (voucher-sales-fleet): Selling was responsive again 2 s after the overload stopped.

Stock reconciliation gap (voucher-sales-fleet): The warehouse recorded 813 vouchers leaving stock but the test received 737 — a difference of 76, i.e. sales the till gave up on that the switch completed anyway.

Requests abandoned on timeout (voucher-sales-fleet): 141 request(s) ran past the 30-second client limit and were given up on.

Recovery after overload (voucher-sales-fleet-2min): Selling was responsive again 21 s after the overload stopped.

Stock reconciliation gap (voucher-sales-fleet-2min): The warehouse recorded 1,885 vouchers leaving stock but the test received 1,642 — a difference of 243, i.e. sales the till gave up on that the switch completed anyway.

Unhandled server errors (voucher-sales-fleet-2min): 68 responses were HTTP 500 with no error message, rather than this API's usual HTTP 200 + errorState + operator-readable text.

Requests abandoned on timeout (voucher-sales-fleet-2min): 212 request(s) ran past the 30-second client limit and were given up on.

Stock reconciliation gap (till-day-in-life): The warehouse recorded 1,814 vouchers leaving stock but the test received 1,807 — a difference of 7, i.e. sales the till gave up on that the switch completed anyway.

Requests abandoned on timeout (till-day-in-life): 8 request(s) ran past the 30-second client limit and were given up on.

6. What was run

Scenario	Result	Outcome
switch-probe	Completed	1,145 requests at 4.9/s, 14.24% errors, worst p95 2,474 ms.
voucher-sales-capacity	Completed	Sold 1,302 vouchers. Selling stayed healthy up to 16 till(s) at 419.7 sales/minute (about 25,182 per hour), and degraded at 32 till(s).
voucher-sales-fleet	Completed	Supports 100 tills, each selling every 20 seconds — 300 sales/minute, about 18,000 an hour. At 150 tills the system could no longer keep up.
voucher-sales-fleet-2min	Limit reached	Supports 700 tills, each selling every 120 seconds — 348.8 sales/minute, about 20,928 an hour. At 900 tills the system could no longer keep up.
till-day-in-life	Completed	Supports at least 800 tills, each selling every 120 seconds — 394.6 sales/minute, about 23,676 an hour. No limit was reached.
release-rollout	Completed	8,185 requests at 15.9/s, 0% errors, worst p95 84,410 ms.
health-baseline	Passed	421 requests at 20/s, 0% errors, worst p95 3 ms.
fleet-poll	Passed	1,772 requests at 16/s, 0% errors, worst p95 7 ms.
cached-feed	Passed	2,818 requests at 55.3/s, 0% errors, worst p95 16 ms.
device-secondary	Passed	358 requests at 10/s, 0% errors, worst p95 14 ms.
update-download	Passed	2 requests at 0/s, 0% errors, worst p95 84,410 ms.

Each scenario's own page on the dashboard carries its full detail: every level of the ramp, per-endpoint latency, individual slow calls, and the server-side series.

People's Prepaid — System Capability Report

lan89 · 11 scenarios · 2026-09-01 16:11 – 12:49

Generated 2026-11-25 11:51 · code version 9b177cf

7. Method and safeguards

- Real sales, not simulated. Selling scenarios perform complete end-to-end purchases against the transaction switch — reserve a request id, then buy — so the latency measured is the latency an operator experiences.
- Demo stock only. Stock is drawn from the buying account's warehouse, so before any sale every account is resolved through the warehouse database (device → branch → customer → warehouse) and the run refuses to start unless all of them trade on the demo warehouse. It fails closed.
- Spend. This campaign consumed 5,819 demo vouchers, leaving 207,656 available.
- Health is judged per level on rates, not raw counts: a level fails on the share of sales that failed, the share of cycles that fell behind, and the time a sale took — never on a single stray event, which at these volumes is background noise.
- Refused is separated from failed. A sale the switch declines (out of stock, no funds) is a stock fact and is never counted against the system; only errors and timeouts are.

8. What the numbers mean

Sales per minute: The best sales-per-minute rate the run held while selling was still healthy — the rate the fleet can be relied on for.

Tills supported: The most tills that were selling at the same time while sales stayed fast and reliable. On a realistic-fleet run this is a supportable FLEET SIZE (tills on a cadence, mostly idle); on a flat-out capacity run it is the number of SIMULTANEOUS sales, which is a much smaller number for the same system.

Served %: How much of the sales the fleet asked for actually completed. 100% means every till got every sale it wanted; the level where this falls away is the number of tills the system can carry.

Late settings syncs: Times a till's 20-second settings-sync cycle was still running when the next was due. This is the point at which a fleet stops being remotely MANAGEABLE — pushed settings and commands start arriving late — and it can arrive before selling itself suffers.

Time to sell one voucher: How long a complete sale took, for 95 out of 100 sales, at the busiest healthy level. This is the wait an operator standing at the till actually feels.

p95: 95 out of 100 requests finished faster than this. When p95 climbs while p50 stays flat, the system is starting to queue under load.

Error rate: The percentage of requests that failed (a bad HTTP status, a business error in the body, a timeout, or a dropped connection).

Recovery time: How long after an overload ended before the system was usable again. It matters twice: an incident outlives the spike that caused it, and a measurement taken before the system has settled will understate its capacity.

Approval

Signing below records that the capability figures above have been reviewed and accepted as the measured capacity of this environment on the dates of the runs.

Reviewed by	Approved by	Date