Take-Home: Control Plane Scheduler (Staff SWE)

Theme: Your mission is to staff our AI agents to call patients throughout the day. Given a CSV of customer call requirements, build a scheduler that outputs **hour-by-hour agent needs**—both **total** and **per customer**—for a single pacific time day. Bonus points for a tiny UI; CLI demo is required.

---

1) Problem (what you're building)

Input: a CSV of requests:

```
None

#CustomerName, AverageCallDurationSeconds, StartTimePT, EndTimePT,
NumberOfCalls, Priority

Stanford Hospital, 300, 9AM, 7PM, 20000, 1
VNS, 120, 6AM, 1PM, 40500, 1
CVS, 180, 11AM, 3PM, 50000, 3
SJC, 1200, 10AM, 12PM, 500, 4
ANMC, 400, 7AM, 8PM, 80000, 5
NMDX, 220, 10AM, 6PM, 40000, 3
```

Output: For each hour 00:00–23:00 PT, print one line:

```
None
00:00 : total=0 ; none
01:00 : total=0 ; none
02:00 : total=0 ; none
03:00 : total=0 ; none
04:00 : total=0 ; none
05:00 : total=0 ; none
06:00 : total=193 ; VNS=193
```

```
07:00 : total=877 ; VNS=193, ANMC=684
08:00 : total=877 ; VNS=193, ANMC=684
09:00 : total=1044 ; Stanford Hospital=167, VNS=193, ANMC=684
10:00 : total=1434 ; Stanford Hospital=167, VNS=193, SJC=84, ANMC=684,
NMDX=306
11:00 : total=2059 ; Stanford Hospital=167, VNS=193, CVS=625, SJC=84,
ANMC=684, NMDX=306
12:00 : total=1975 ; Stanford Hospital=167, VNS=193, CVS=625, ANMC=684,
NMDX=306
13:00 : total=1782 ; Stanford Hospital=167, CVS=625, ANMC=684, NMDX=306
14:00 : total=1782 ; Stanford Hospital=167, CVS=625, ANMC=684, NMDX=306
15:00 : total=1157 ; Stanford Hospital=167, ANMC=684, NMDX=306
16:00 : total=1157 ; Stanford Hospital=167, ANMC=684, NMDX=306
17:00 : total=1157 ; Stanford Hospital=167, ANMC=684, NMDX=306
18:00 : total=851 ; Stanford Hospital=167, ANMC=684
19:00 : total=684 ; ANMC=684
20:00 : total=0 ; none
21:00 : total=0 ; none
22:00 : total=0 ; none
23:00 : total=0 ; none
```

Baseline assumptions (keep it simple):

- Time buckets are **hourly**, **StartTime inclusive**, **EndTime exclusive** (e.g., 9AM–7PM = 09:00–18:59 bucketed as 09–18).
- Calls are **uniformly distributed** across the active hours.
- **Agents needed per hour per customer** = ceil(calls_per_hour * avg_call_duration_seconds / 3600).
- **Total** is the sum of per-customer agents for that hour.

You may document and implement different assumptions (e.g., <100% utilization, priority preemption) if you state them clearly and keep the CLI compatible.

---

2) What to build (MVP)

- A **CLI tool** (any mainstream language) that:
    - **Reads** the input CSV (path passed as an argument).
    - **Validates** rows (time parsing, numeric fields, priority range 1–5).
    - **Computes** agents needed per hour per customer and totals.
    - **Prints** exactly 24 lines as specified above.
    - **Option:** --utilization <0..1> multiplier to size agents more conservatively.
- **Design doc** (≤ 1 page) covering:
    - High-level architecture (modules, data flow).
    - Core algorithms & key trade-offs.
    - Observability & testing approach.
    - Future enhancements (see §5).

Nice bonus: A tiny **UI** that shows a 24-cell grid with totals and per-customer breakdown on hover/click.

---

3) How to run (demo)

- Provide a **Makefile** or simple commands:
    - make run INPUT=./input.csv
    - make test
- CLI should accept:
    - --input <path> (required)
    - --utilization <float> (optional; default 1.0)

    - --format text|json|csv (optional; default text)
- Print to stdout; if --format=json, emit a structure like:

4) Quality bar (what we'll look for)

- **Correctness:** sane parsing, edge cases (12AM/PM, end before start, empty rows).
- **Clarity:** readable code, small functions, comments where it counts.
- **Tests:** unit tests for parsing and the hourly math; 1 E2E golden test.
- **DX:** a clean README and deterministic output.
- **Design judgment:** simple, extensible boundaries (parser, calculator, formatter).

---

5) Stretch ideas (pick 1–2 if time allows)

- **Priority-aware smoothing:** if number of agents (capacity) is given as input constraint, allocate hours by priority (1 highest) to meet the demand. Output details on specific hours when demands will not be met and which clients will be impacted.
- **Calendar:** Build the input csv through a Calendar like UI
- **Multiple Shifts:** allow a client to specify demand for follow up calls (time/volume).
- **Observability:** basic metrics (e.g., total calls ingested, compute time, unmet demand).
- **Timezone & daylight saving:** robust handling with a library.

---

7) Sample acceptance checks

- **Exactly 24 lines** printed for --format=text.

- For each customer, agent counts appear **only** within its active window.
- **Idempotent**: running twice with same inputs yields identical output.
- **Golden test**: provided sample CSV produces a stable JSON result (commit a golden file).

---

8) Submission

- Git repo (zip or link) with source, tests, README, and sample CSV.
- Include your **design doc (≤1 page)** and a **short note**: what you'd do next with 1 more week.

Time guide: 3–5 focused hours for MVP; stretch goals optional.

Have fun—make the output satisfying to read, and the code pleasant to change!