Instawork

QA take-home assignment

The problem you'll solve

Instawork keeps every manual test case as a JSON file in a Git repository.
Any time Product ships a new feature—or tweaks / fixes an old one — someone has to:

1. figure out which existing test cases should change,
2. write brand-new cases for the uncovered behaviour, and
3. document what changed and why.

It's slow and error-prone.
We want a command-line tool that lets a QA engineer drop in a plain-English change request and receive:

- updated JSON files for impacted tests,
- new JSON files for uncovered scenarios, and
- a short report explaining the decisions.

Supplied assets

The starter repo you'll receive contains:

```
None
IW_OVERVIEW.md            # IW products background for LLM
context

test_cases/               # existing test cases

schema/                   # JSON-schema for test cases

sample_change_requests/   # example change requests
```

Check out the attached zip in the email to download the assets.

Instawork

What you will build — from the QA-engineer’s perspective

When a QA engineer runs the tool, it should:

1. Understand context
   • Read `IW_OVERVIEW.md` automatically.
   • Accept a change-request file or string that includes:
   – `change_type`: `new_feature`, `feature_update`, or `bug_fix`
   – acceptance criteria / user flows written in plain English.
2. Figure out impact
   • Determine which existing test cases (if any) are affected by the request.
   • Decide whether each affected case needs an update.
3. Generate output
   • Update only the necessary fields of impacted cases while keeping the JSON schema valid.
   • Add at least three new test cases—*positive*, *negative*, and *edge*—when new behaviour is introduced.
   • Write all new or modified cases back to `test_cases/`.
   • Produce a human-readable `report.md` that lists:
   – which cases were touched and why,
   – which brand-new cases were added and why,
   – any assumptions or open questions for the team.

That’s it — the *how* is up to your creativity.

Functional checklist

- Ingest `IW_OVERVIEW.md` and the change request automatically.
- Identify and update impacted test cases.
- Generate ≥ 3 new cases when required (`positive`, `negative`, `edge`).
- Ensure every written file conforms to `schema/test_case.schema.json`.
- Emit `report.md` as described above.

Non-functional expectations

- Language: Python or JavaScript/TypeScript preferred (use another if you wish).
- Use at least one LLM available to you.
- Provide unit tests for your core logic.

- The project should work on macOS/Linux.
- Docker for easy setup and execution is a bonus.

Guidance

You are free to design the internals as you see fit. Techniques people often use include:

- vector or keyword search to find related test cases,
- an LLM call that drafts JSON adhering to the schema,
- a quick schema-validation pass before writing to disk.

What to submit

1. Create a **public** GitHub repository containing:
   - `src/` – your code
   - `tests/` – automated unit tests
   - `README.md` – how to run it, architecture choices, future work
2. Invite the following github users to your project
   - antonyfuentesdev
   - deeptypatel
   - dimbaser
   - ovalerio280788
   - Sharathannaiah
   - theUnrelated
3. *Share a voice over screen recording* ≤ 7 minutes giving a demo of your solution, along with your thought process and approach.

Note: Aim for optimal scalability, ensuring it can effectively manage a TC repository of over 1000 test cases. Please don't over-engineer; ship, document gaps, and outline next steps.

Good luck—and have fun!