Here's the tight SOW snapshot for **Cubism Project V2.3.7** as of today

.

1) Status snapshot

1. Repo: `cubist_art` on Google Drive. Working copy on Dell G7. Secondary copy on Alienware.

2. Python: Dell is running 3.13.7. Alienware likely used 3.12. Virtual env lives inside the synced project folder.

3. Tests: Pytest "collects 11 items" but runs 0 because a module import fails during collection.

4. Primary blocker: tests (and possibly fixtures/CLI) expect `run_cubist` in `cubist_core_logic`, but that symbol is not present. The environment itself is healthy enough to import the package and run pytest, but the public API changed.

5. Tooling: Added `tools/dev_diag.ps1` for one-shot diagnostics. Provided a merged `.vscode/tasks.json` that adds "Tests: Pytest (log to .diag/pytest_last.txt)" and a diagnostics task. You ran `dev_diag` successfully; it confirmed the import failure.

2) Known problems

1. **Missing public API symbol**: `run_cubist` no longer exists in `cubist_core_logic`. Tests or fixtures still import it, causing import-time failure and zero tests executed.

2. **Version drift**: 3.13.7 on Dell vs 3.12 on Alienware. Some dependencies and your code path were validated on 3.12, not 3.13.

3. **Venv in Google Drive**: The virtual environment lives under a synced path. That invites file locks, sync races, and path issues on multi-machine setups.

4. **GUI gaps** (from prior run): missing log text box, progress bar not updating, some geometry options not wired in the UI.

5. **Output gaps**: pipeline reported "No SVG for None …" messages earlier. That suggests geometry steps ran without producing artifacts or the output path was not set correctly.

6. **Plugin test fragility**: a prior geometry plugin discovery test had failed on the old machine for concentric circles reproducibility. This likely needs a quick recheck once tests run again.

7. **Path edge cases**: spaces in `G:\My Drive\...` are OK, but any ad-hoc shell scripts or brittle path joins may fail. Keep using quoted paths in tasks and scripts.

8. **Inconsistent import surface**: callers import from different modules (`cubist_core_logic`, CLI, registry). Without a single stable facade, refactors break tests and tools.

3) Completed or in place

1. **Diagnostics**: `tools/dev_diag.ps1` runs a standard set of checks and tees pytest output to `.diag/pytest_last.txt`.

2. **VS Code tasks**: merged `tasks.json` provided to run tests with logging and to invoke the diagnostics script easily.

3. **Actionable triage**: PowerShell-safe one-liners to inspect what the module exports and to grep for `run_cubist` references.

4. **Back-compat plan**: a safe, reversible **shim** to restore `run_cubist` by delegating to the current entrypoint (`run_pipeline`). Script provided to append it with a

backup (`tools/add_run_cubist_shim.ps1`). Not yet run.

4) Open tasks and priorities

P0 — unblock the test runner

1. **Restore public API**. Choose one:
    - A) Append the `run_cubist` shim to `cubist_core_logic.py` using `tools/add_run_cubist_shim.ps1`. This keeps existing tests stable and buys time.
    - B) Update tests and any fixtures to import the current entrypoint (for example `cubist_cli.run_pipeline`). This is cleaner long-term.
      Acceptance: `dev_diag` shows a clean import of `cubist_core_logic`. `pytest` executes all collected tests (pass/fail/skip is fine, but not zero).
2. **Stabilize Python version**. Pin project to 3.12 for v2.3.7.
   Acceptance: `py -3.12 -m venv C:\venvs\cubist_art-3.12` works, requirements install cleanly, pytest runs under this interpreter.
3. **Move venv outside Drive**. Keep source under Google Drive, put the venv in `C:\venvs\...` and add that Scripts folder to PATH in tasks.
   Acceptance: VS Code tasks run `python` from the external venv without needing manual activation.

P1 — make the pipeline and UI solid

4. **SVG output regression**. Re-run the CLI pipeline after tests execute and confirm SVGs are written to the expected `output/production/...` paths.
   Acceptance: each geometry selected produces `.svg` (and any other expected assets) under the timestamped run folder. No “No SVG for None” messages.

5. **GUI fixes**. Ensure the log text box renders and is fed by your logger. Make the progress bar advance per geometry and per stage.
   Acceptance: visible log stream during runs; progress bar reflects geometry steps and completes to 100% when done.

6. **Plugin pass**. Validate all registered geometries (voronoi, delaunay, poisson, scatter circles, rectangles, cascade, concentric circles). Confirm deterministic output for a fixed seed.
   Acceptance: plugin discovery returns callables; for a fixed seed and same input image, geometry calls yield deterministic metrics or hashes in a smoke test.

7. **Stable import facade**. Add `cubist_api.py` that re-exports the supported surface, and point tests and CLI helpers to it. Keep `cubist_core_logic` free to evolve.
   Acceptance: all in-repo imports use `cubist_api` for the public functions. A refactor of internals does not break tests.

P2 — quality of life and guardrails

8. **Editable install**. Add a minimal `pyproject.toml` and run `pip install -e .` in the venv.
   Acceptance: imports resolve without relying on `PYTHONPATH=.` hacks.

9. **Pre-commit checks**. Add a local pre-commit or simple `tools/preflight.ps1` that runs ruff, mypy (if used), and pytest smoke before commits.
   Acceptance: one command reports style, types, and a fast test pass.

10. **Reduce handoff friction**. Wire the **Continue** VS Code extension to your OpenAI key so you can chat in-repo without copy/paste. Keep the `.diag` folder for quick logs.
    Acceptance: you can run a test task, then say “open .diag/pytest_last.txt” in-IDE and get immediate help.

5) Decisions needed from you

1. Pick the P0 path: apply the **shim** now, or update tests to the current entrypoint.

2. Confirm we can pin to **Python 3.12** for v2.3.7 and create the venv under `C:\venvs\cubist_art-3.12`.

3. Confirm replacing `.vscode/tasks.json` with the merged version I provided.

4. Give the green light to add `cubist_api.py` and migrate test imports once the blocker is cleared.

5. Confirm you want the Continue integration steps added to the repo docs.

6) Immediate next actions (my recommendation)

1. Run: `pwsh -File tools/add_run_cubist_shim.ps1`.

2. Run: `Terminal → Run Task… → Tests: Pytest (log to .diag/pytest_last.txt)` and send me the new log.

3. If tests execute, re-run the CLI pipeline once to check for SVGs and GUI logging. Report any geometry that still emits “No SVG for None”.

If you prefer, I can ship full-file replacements for `cubist_core_logic.py` and a new `cubist_api.py` that harden the surface right away.