

Code that survives the checks.

Explore repository tasks with a runnable starting state, an explicit output contract, and deterministic checks.

Task and inputs

Refactor a small service without changing its public behavior; submit the patch and a structured verification report.

A fixed repository revision, setup instructions, representative fixtures, dependency constraints, and the required report schema. State which files may change and which commands establish the baseline.

Working steps

1. Recreate the starting environment, run the baseline checks, and map the code paths affected by the request.
2. Make the smallest coherent change. Preserve interfaces and record any assumptions needed to interpret incomplete instructions.
3. Run the agreed checks, inspect the diff, and produce a report connecting each requirement to evidence.

Deliverables

- Task workspace
- Code patch
- Test results
- Review notes

Review and acceptance

Check behavior, output structure, and task constraints against a fixed starting revision.

- The workspace starts cleanly and the verification commands can be repeated.
- Required outputs satisfy their schema; failed checks remain visible in the report.
- The patch respects allowed paths, dependencies, and behavior outside the requested change.

Delivery boundary

Acceptance covers the specified fixtures and environment. Broader performance, security, and production compatibility require separately agreed checks; passing the example does not establish them.

This guide describes a project approach. It is not a customer result, a performance benchmark, or a statement of certification.

Related guide: [/programs/coding](#) | Browse downloadable demonstrations at [/samples](#).