STUNIR Code Quality Checklist

Use this checklist when reviewing or submitting code.

Before Committing

General

- [ ] Code compiles/runs without errors
- [ ] All tests pass
- [ ] Pre-commit hooks pass
- [ ] No debug/print statements left in code
- [ ] No commented-out code
- [ ] No hardcoded secrets or credentials

Python

- [ ] Black formatting applied
- [ ] isort import ordering
- [ ] Ruff linting passes
- [ ] MyPy type checking passes (no new errors)
- [ ] Docstrings on all public functions/classes
- [ ] Type hints on function signatures
- [ ] No bare `except:` clauses
- [ ] Tests added for new functionality

Rust

- [ ] `cargo fmt` applied
- [ ] `cargo clippy` passes
- [ ] No `unsafe` code without justification comment
- [ ] rustdoc comments on public items
- [ ] Tests added for new functionality
- [ ] `cargo test` passes

Haskell

- [ ] hlint suggestions addressed
- [ ] Compiles with `-Wall -Werror`
- [ ] Haddock comments on exports
- [ ] Type signatures on all functions
- [ ] Tests added for new functionality

Code Review Checklist

Architecture

- [ ] Changes fit the module organization
- [ ] No circular dependencies introduced

- [ ] Public/private boundaries respected
- [ ] API versioning considered for breaking changes

Security

- [ ] Input validation present
- [ ] No SQL injection risks
- [ ] No path traversal vulnerabilities
- [ ] Cryptographic operations use standard libraries
- [ ] No sensitive data in logs

Performance

- [ ] No unnecessary allocations in hot paths
- [ ] Algorithms have appropriate complexity
- [ ] Large data handled efficiently
- [ ] No blocking operations in async code

Determinism (STUNIR-specific)

- [ ] JSON output is canonicalized (sorted keys)
- [ ] Timestamps use epoch integers, not formatted strings
- [ ] File operations produce same output across platforms
- [ ] No reliance on dictionary ordering (Python <3.7)
- [ ] Hash computations are reproducible

Documentation

- [ ] README updated if needed
- [ ] API documentation updated
- [ ] Changelog entry added
- [ ] Migration guide for breaking changes

Testing

- [ ] Unit tests for new functions
- [ ] Edge cases covered
- [ ] Error conditions tested
- [ ] Determinism tests for output consistency
- [ ] Integration tests for workflows

Coverage Targets

Component	Target	Current
Python Core	80%	-
Python Manifests	80%	-
Rust Native	75%	-
Haskell Native	70%	-

Quality Metrics

Ruff Rules Enabled

- E: pycodestyle errors
- W: pycodestyle warnings
- F: Pyflakes
- I: isort
- N: pep8-naming
- D: pydocstyle
- UP: pyupgrade
- B: flake8-bugbear
- C4: flake8-comprehensions
- SIM: flake8-simplify
- S: flake8-bandit (security)
- PTH: flake8-use-pathlib
- PL: Pylint
- PERF: Perflint

Clippy Lints (Rust)

- `pedantic`: Enabled as warnings
- `nursery`: Enabled as warnings
- `suspicious`: Denied
- `perf`: Enabled as warnings
- `unsafe_code`: Denied

GHC Warnings (Haskell)

- `-Wall`: All standard warnings
- `-Wcompat`: Future compatibility
- `-Wincomplete-patterns`: Pattern matching
- `-Wmissing-export-lists`: Export control
- `-Wredundant-constraints`: Type constraints