/sentry-code-review

Source: `~/.claude/skills/sentry-code-review/SKILL.md`

---

name: code-review description: Perform code reviews following Sentry engineering practices. Use when reviewing pull requests, examining code changes, or providing feedback on code quality. Covers security, performance, testing, and design review.

Sentry Code Review

Follow these guidelines when reviewing code for Sentry projects.

Review Checklist

Identifying Problems

Look for these issues in code changes:

- **Runtime errors**: Potential exceptions, null pointer issues, out-of-bounds access

- **Performance**: Unbounded O(n²) operations, N+1 queries, unnecessary allocations
- **Side effects**: Unintended behavioral changes affecting other components
- **Backwards compatibility**: Breaking API changes without migration path
- **ORM queries**: Complex Django ORM with unexpected query performance
- **Security vulnerabilities**: Injection, XSS, access control gaps, secrets exposure

Design Assessment

- Do component interactions make logical sense?
- Does the change align with existing project architecture?
- Are there conflicts with current requirements or goals?

Test Coverage

Every PR should have appropriate test coverage:

- Functional tests for business logic
- Integration tests for component interactions
- End-to-end tests for critical user paths

Verify tests cover actual requirements and edge cases. Avoid excessive branching or looping in test code.

Long-Term Impact

Flag for senior engineer review when changes involve:

- Database schema modifications
- API contract changes
- New framework or library adoption
- Performance-critical code paths
- Security-sensitive functionality

Feedback Guidelines

Tone

- Be polite and empathetic
- Provide actionable suggestions, not vague criticism
- Phrase as questions when uncertain: "Have you considered...?"

Approval

- Approve when only minor issues remain
- Don't block PRs for stylistic preferences
- Remember: the goal is risk reduction, not perfect code

Common Patterns to Flag

Python/Django

```
# Bad: N+1 query
for user in users:
    print(user.profile.name)  # Separate query per user

# Good: Prefetch related
users = User.objects.prefetch_related('profile')
```

TypeScript/React

```
// Bad: Missing dependency in useEffect
useEffect(() => {
  fetchData(userId);
}, []);  // userId not in deps

// Good: Include all dependencies
useEffect(() => {
  fetchData(userId);
}, [userId]);
```

Security

```
# Bad: SQL injection risk
cursor.execute(f"SELECT * FROM users WHERE id = {user_id}")

# Good: Parameterized query
cursor.execute("SELECT * FROM users WHERE id = %s", [user_id])
```

References

- Sentry Code Review Guidelines

Revision #4
Created 2026-02-18 08:39:58 UTC by John
Updated 2026-05-31 20:01:29 UTC by John