**AI Code Review System Complete Solution**

**Problem Decomposition**1. PR\_Analysis\_Agent:

- Input: PR metadata, code diff

- Output: Change summary, affected files

- Success: Accurate change identification

- Failure: Fallback to manual review

2. Static\_Analysis\_Agent:

- Input: Code changes

- Output: Security issues, code smells

- Success: 90% vulnerability detection

- Failure: Flag for human review

3. AI\_Code\_Review\_Agent:

- Input: Code + context

- Output: Review comments, suggestions

- Success: <4 hour review time

- Failure: Escalate to senior dev

4. Test\_Impact\_Analysis\_Agent:

- Input: Code changes + test suite

- Output: Affected tests, new test needs

- Success: Identify broken tests

- Failure: Run full test suite

5. Deployment\_Readiness\_Agent:

- Input: All previous outputs

- Output: Go/No-go decision

- Success: Correct deployment decision

- Failure: Conservative (No-go)  
  
  
**Parallel vs Blocking:**Parallel: Static Analysis ↔ Test Impact Analysis

Blocking: AI Review (depends on analysis completion)

Critical Decision: Deployment Readiness (final gate)  
  
**Handoff Points:**PR Analysis → All agents: Change summary + context

Static Analysis → AI Review: Security findings + metrics

All agents → Deployment: Consolidated report + confidence scores

**AI Prompting Strategy**

**Sample Prompts:**

**Prompt 1: Code Review Agent**

SYSTEM\_PROMPT = """

You are a senior software engineer with 10+ years experience.

Review this pull request focusing on:

CRITICAL AREAS:

1. Code quality & best practices

2. Potential bugs & edge cases

3. Security vulnerabilities (OWASP Top 10)

4. Performance implications

5. Maintainability & readability

INPUT FORMAT:

{

"code\_changes": [file\_diffs],

"pr\_description": "string",

"repository\_context": "tech\_stack, patterns",

"previous\_reviews": [past\_comments]

}

OUTPUT FORMAT:

{

"overall\_assessment": "APPROVE|REQUEST\_CHANGES|COMMENT",

"critical\_issues": [

{

"file": "filename",

"line": 123,

"issue": "specific problem",

"severity": "HIGH|MEDIUM|LOW",

"suggestion": "concrete fix"

}

],

"suggestions": [

{

"file": "filename",

"line": 456,

"suggestion": "improvement",

"priority": "HIGH|MEDIUM|LOW"

}

],

"questions": ["clarification questions for author"],

"confidence\_score": 0.95

}

EXAMPLES:

GOOD: "Found SQL injection vulnerability in user\_id parameter - use parameterized queries"

BAD: "Looks good to me" (too vague)

"""  
  
**Prompt 2: Security Review Agent**

SYSTEM\_PROMPT = """

You are a security specialist focusing on application security.

Identify security vulnerabilities using OWASP Top 10 framework.

SPECIFIC CHECKS:

- Injection vulnerabilities (SQL, NoSQL, Command)

- Broken authentication & session management

- Sensitive data exposure

- XML external entities (XXE)

- Broken access control

- Security misconfigurations

- Cross-site scripting (XSS)

- Insecure deserialization

- Using components with known vulnerabilities

- Insufficient logging & monitoring

ERROR HANDLING:

- If unfamiliar with library/framework, flag for expert review

- When uncertain, prioritize security (conservative approach)

- Provide specific remediation steps for each finding

"""

**Challenging Scenarios Handling:**

# Obscure Libraries:

"Focus on general security patterns. If library is unfamiliar, check for:

- Input validation boundaries

- Output encoding requirements

- Authentication/authorization flows

- Data serialization/deserialization"

# Security Reviews:

"Apply defense-in-depth principles:

1. Input validation at boundaries

2. Output encoding for context

3. Principle of least privilege

4. Secure default configurations"

# Performance Analysis:

"Analyze for:

- N+1 query patterns

- Missing database indexes

- Inefficient algorithms (O(n^2) vs O(n log n))

- Memory leaks & resource management"

# Legacy Code:

"Risk-based approach:

- Identify critical security fixes

- Suggest incremental improvements

- Focus on attack surface reduction

- Document technical debt"

**Ensuring Prompt Effectiveness:**

validation\_strategy = {

"golden\_dataset": "Test prompts against known good/bad code samples",

"multi\_llm\_consensus": "Use 2-3 different LLMs and compare results",

"false\_positive\_tracking": "Monitor and tune based on FP/FN rates",

"developer\_feedback": "Incorporate real developer feedback into prompt tuning",

"regular\_evaluation": "Monthly review of prompt effectiveness metrics"

}

**---------------System Architecture & Reusability--------------------**

**Making System Reusable:**

configuration\_template = {

"team\_specific\_rules": {

"backend\_team": {"focus\_areas": ["api\_security", "database\_performance"]},

"frontend\_team": {"focus\_areas": ["xss\_prevention", "ui\_performance"]},

"mobile\_team": {"focus\_areas": ["data\_storage", "network\_security"]}

},

"language\_support": {

"python": {"tools": ["pylint", "bandit", "safety"]},

"javascript": {"tools": ["eslint", "npm\_audit"]},

"java": {"tools": ["spotbugs", "dependency\_check"]}

},

"deployment\_targets": {

"aws": {"services": ["CodeDeploy", "ECS", "Lambda"]},

"azure": {"services": ["Azure DevOps", "AKS"]},

"on\_prem": {"services": ["Jenkins", "Kubernetes"]}

}

}  
  
**Continuous Improvement:**

learning\_framework = {

"false\_positive\_analysis": {

"track\_patterns": "Common FP patterns across teams",

"adjust\_thresholds": "Tune sensitivity based on FP rates",

"update\_prompts": "Refine AI prompts based on FP analysis"

},

"deployment\_correlation": {

"success\_patterns": "What review comments correlate with successful deployments?",

"failure\_patterns": "What missed issues cause production incidents?",

"feedback\_loop": "Use incident data to improve review criteria"

},

"developer\_feedback\_incorporation": {

"rating\_system": "Developers rate review quality",

"comment\_effectiveness": "Which suggestions are actually implemented?",

"preference\_learning": "Learn team-specific review preferences"

}

}

**Implementation Strategy**

**Month Roadmap:**

roadmap = {

"Month 1-2: MVP": [

"Basic PR analysis integration",

"Simple static analysis (existing tools)",

"GitHub/GitLab webhook setup",

"Basic reporting dashboard"

],

"Month 3-4: Enhanced Features": [

"AI code review integration",

"Test impact analysis",

"Basic deployment gates",

"Team-specific configurations"

],

"Month 5-6: Advanced Capabilities": [

"Automated rollback triggers",

"Multi-environment support",

"Advanced analytics & reporting",

"Self-learning improvements"

]

}

**Risk Mitigation:**

risk\_plan = {

"ai\_incorrect\_decisions": {

"human\_fallback": "Critical changes always require human review",

"confidence\_thresholds": "Only auto-approve high-confidence reviews",

"gradual\_rollout": "Start with non-critical repositories"

},

"system\_downtime": {

"fallback\_mode": "Revert to manual process during outages",

"circuit\_breakers": "Fail open for critical deployments",

"monitoring": "Real-time alerting for system health"

},

"team\_resistance": {

"opt\_in\_phases": "Teams can choose when to adopt",

"training\_program": "Comprehensive onboarding",

"success\_metrics": "Show tangible benefits (time saved, quality improved)"

}

}

**Tool Integration:**

tool\_ecosystem = {

"code\_review\_platforms": ["GitHub", "GitLab", "Bitbucket"],

"ci\_cd\_systems": ["Jenkins", "GitHub Actions", "GitLab CI", "CircleCI"],

"monitoring\_tools": ["Datadog", "New Relic", "Prometheus", "Grafana"],

"security\_scanners": ["SonarQube", "Snyk", "Veracode", "Checkmarx"],

"communication\_tools": ["Slack", "Microsoft Teams", "Jira", "Confluence"]

}