Telomere

AI Agentic Solution for SDLC Workflow

MVP Project Requirement Document (MVP-PRD)

---

1. PROJECT OBJECTIVE

Build an **AI Agentic Solution** that:

- Monitors project tickets in **Jira**
- Gathers relevant documentation from **Confluence**
- Analyzes the codebase in **GitHub**
- Generates:
    - A structured implementation plan
    - Proposed code changes
    - Suggested unit tests
    - Documentation updates
    - A draft Pull Request for human review

The application is deployed on **Amazon Web Services**.

All AI processing must remain inside the AWS VPC.

Human review is mandatory before any staging deployment.

---

2. PROBLEM STATEMENT

Currently:

- Product Owners and QA submit bug or feature tickets in Jira.
- Documentation exists in Confluence.
- Source code resides in GitHub.
- Developers manually interpret tickets, review documentation, and plan implementation.

Challenges:

- Tickets often lack sufficient detail.
- Developers spend time clarifying requirements.

- Context switching between Jira, Confluence, and GitHub slows execution.

The goal is to create an AI assistant that reduces interpretation time and prepares a structured development proposal.

---

3. FUNCTIONAL REQUIREMENTS

3.1 Ticket Monitoring

The system must:

- Monitor a specific Jira project.
- Trigger when:
  - A new ticket is created, or
  - A ticket moves to a “Ready for Dev” status.
- Extract ticket details (title, description, attachments, reporter, labels).

3.2 Ticket Completeness Check

The system must:

- Evaluate whether the ticket contains enough implementation detail.
- Detect missing acceptance criteria or unclear expected behavior.

If information is insufficient:

1. Post a structured comment in Jira:
   - Summarize the AI’s understanding.
   - List assumptions.
   - Ask specific clarification questions.
2. Pause further processing until confirmation.

The system must not proceed with ambiguous requirements.

3.3 Context Retrieval

The system must retrieve relevant information from:

- Confluence documentation (architecture, business logic, API notes).
- GitHub repository (related modules, existing patterns, tests).

Basic semantic search or keyword matching is acceptable for MVP.

3.4 Implementation Package Generation

For sufficiently defined tickets, the system must generate:

A. Implementation Plan

- Summary of requested change.
- Impacted components.
- Deployment considerations.
- Risk level (Low/Medium/High).
- Confidence score.

B. Proposed Code Changes

- Suggested file modifications (diff-style or structured explanation).
- Clear explanation of rationale.

C. Suggested Unit Tests

- Recommended test cases.
- Edge cases.
- Validation logic.

D. Documentation Update Suggestions

- Identify which Confluence pages may need updates.

3.5 Pull Request Creation

The system must:

- Create a Draft Pull Request in GitHub.
- Link it to the Jira ticket.
- Include:

    - Implementation plan
    - Code proposal
    - Test suggestions
    - AI-generated disclaimer
- Assign to a human developer for review.

No automatic merge or deployment.

---

4. NON-FUNCTIONAL REQUIREMENTS

Security

- All processing must remain inside AWS VPC.
- No code or documentation may leave AWS environment.
- Secure credential handling.

Governance

- No automatic production deployment.
- All PRs must require human approval.
- Log AI decisions and actions.

Performance

- Must support 60–100 tickets per month.
- Processing time per ticket should be reasonable (target under 10 minutes).