Requirements Document — Project Delivery & Maintenance Software

# 1. Introduction

This system is intended to streamline project delivery, release management, and long-term maintenance by integrating Agile project management, DevOps automation, and cloud-agnostic deployment. It must support collaborative teams, scalable workflows, and compliance requirements.

# 2. Objectives

* Provide a single platform for project planning, delivery, and support.
* Enable Agile ways of working with Scrum/Kanban boards, sprint tracking, velocity insights.
* Support DevOps practices: CI/CD, test automation, monitoring, infrastructure-as-code.
* Ensure vendor-neutral cloud support (deployable to AWS, Azure, GCP, on-prem).
* Integrate seamlessly with JIRA, GitHub/GitLab/Bitbucket, Jenkins/GitHub Actions/Azure DevOps.
* Improve traceability from requirements → development → deployment → maintenance.

# 3. Functional Requirements

## 3.1 Project & Agile Management

* Backlog management (epics, user stories, tasks, bugs).
* Sprint planning & tracking (capacity planning, burndown).
* Kanban workflows for continuous delivery teams.
* Integration with JIRA for real-time synchronization.
* Role-based dashboards (Project Manager, Product Owner, Developer, QA, Ops).
* Reporting: Velocity charts, cycle time, lead time, throughput.

## 3.2 DevOps & CI/CD

* Integration with CI/CD pipelines (Jenkins, GitHub Actions, GitLab CI, Azure DevOps).
* Automated build, test, deploy workflows.
* Environment-aware deployments (dev, test, staging, prod).
* Container orchestration (Docker, Kubernetes).
* Artifact management (Nexus, JFrog Artifactory).
* Monitoring integration (Prometheus, Grafana, ELK).

## 3.3 Cloud Agnostic Deployment

* Support for AWS, Azure, GCP, On-premises via Terraform/Helm/Ansible.
* Pluggable deployment templates.
* Abstracted cloud service definitions (storage, compute, networking).
* Automated scaling and cost-optimization policies.

## 3.4 Maintenance & Operations

* Incident management integration (JIRA Service Management, ServiceNow, PagerDuty).
* Automated SLA tracking.
* Knowledge base & documentation repository.
* Change management workflows.
* Post-release health checks & automated rollbacks.

# 4. Non-Functional Requirements

* Scalability: Handle 1000+ concurrent users.
* Performance: <2s latency for dashboard updates.
* Security: RBAC, SSO (SAML/OAuth2), audit logging, data encryption (at rest & transit).
* Compliance: GDPR, SOC2, ISO27001 alignment.
* Reliability: 99.9% uptime SLA.
* Extensibility: API-first, plugin support.

# 5. Agile & DevOps Integration

The solution must enable Agile practices (Scrum, Kanban) integrated with DevOps workflows. User stories and backlog items must link directly to code commits, CI/CD pipelines, and deployment artifacts. Dashboards should visualize sprint health, deployment frequency, lead time for changes, and mean time to recovery (MTTR).

# 6. User Stories

## Project Manager

* As a Project Manager, I want to create and manage project backlogs, so that I can prioritize work effectively.
* As a Project Manager, I want real-time JIRA synchronization, so that I can avoid duplication of effort.

## Product Owner

* As a Product Owner, I want to define epics and user stories, so that I can align work with business value.
* As a Product Owner, I want reports on velocity and cycle time, so that I can track progress.

## Developer

* As a Developer, I want automated CI/CD pipelines, so that I can focus on coding instead of deployments.
* As a Developer, I want integration with Git repositories, so that I can manage source code easily.

## QA Engineer

* As a QA Engineer, I want automated testing in pipelines, so that I can ensure code quality.
* As a QA Engineer, I want dashboards for defects and test coverage, so that I can monitor quality metrics.

## Ops Engineer

* As an Ops Engineer, I want cloud-agnostic deployment support, so that I can manage environments easily.
* As an Ops Engineer, I want monitoring and alerts integration, so that I can respond to incidents quickly.

# 7. Acceptance Criteria

* All user stories must be testable with clear definition of done.
* System must integrate with at least one tool per integration category (JIRA, Git, CI/CD, Monitoring).
* CI/CD pipelines must support automated tests, build, and deployments.
* Cloud deployment must be validated on AWS, Azure, and GCP using IaC.
* Dashboards must update in real-time with <2s latency.
* Security requirements (RBAC, SSO, encryption) must be enforced.