# 3. Requirements Gathering

## Stakeholder Analysis

Key Stakeholders:  
- Developers: Need a streamlined CI/CD process to reduce manual work and improve productivity.  
- DevOps Engineers: Require robust automation tools to maintain deployment pipelines efficiently.  
- QA Testers: Need automated testing frameworks to validate software quality consistently(using Jenkins framework).  
- Project Managers: Require insights into deployment frequency, success rates, and project progress.  
- End Users: Expect a reliable and responsive application with minimal downtime.

## User Stories & Use Cases

User Stories:  
- As a developer, I want to push code to a Git repository and trigger automated builds and deployments.  
- As a DevOps engineer, I want to monitor the CI/CD pipeline and get alerts on build failures.  
- As a QA tester, I want automated tests to run with every deployment to ensure code quality.  
- As a project manager, I want to view deployment metrics and pipeline performance.  
  
Use Cases:  
1. Code Commit and CI/CD Trigger:  
 - Developer pushes code to Git -> Jenkins pipeline triggers -> Maven builds the application -> Docker containers are created -> Kubernetes deploys to AWS.  
2. Automated Testing Scenario:  
 - Code is committed -> Jenkins triggers test stage -> Test results are logged -> QA team reviews test outcomes.  
3. Application Deployment:  
 - New feature is ready -> Pipeline promotes to production -> Kubernetes performs a rolling update -> Application is live without downtime.

## Functional Requirements

- Source Code Management using Git.  
- Continuous Integration and Continuous Deployment (CI/CD) with Jenkins.  
- Build Automation using Maven.  
- Containerization of application using Docker.  
- Container Orchestration with Kubernetes.  
- Cloud Deployment on AWS (EKS, EC2, S3).  
- Automated Testing Integration (e.g., Jenkins).  
- Monitoring and Alerting for application health and performance.

## Non-functional Requirements

- Performance: Application should handle 1000 concurrent users with a response time of under 2 seconds.  
- Security: Implement role-based access control in Jenkins and Kubernetes.  
- Usability: CI/CD pipeline should have clear logs and notifications for stakeholders.  
- Reliability: Achieve 99.9% application uptime using Kubernetes' self-healing and load balancing features.  
- Scalability: Support auto-scaling of application instances based on traffic using Kubernetes Horizontal Pod Autoscaler.