**DESIGN AND ARCHITECTURE DOCUMENT**

**JAVA MUTATION RESEARCH PROJECT:**

This document provides information about the design and architecture of the Java Mutation Project along with the deliverables shared upon the completion of the project.

March 11, 20xx

Version 1.0

ABC Solutions

**Project Team**

| Team Member | Role | Responsibilities |
| --- | --- | --- |
|  | Product Owner | Project Information, Project Team, Scope and Definitions, Project Diagram  Communicates product goals and backlog items, final decision maker, and represents the needs of the client. |
|  | Scrum Master | Use Case Diagram View  Coaches the team in self-management, helps create high-value increments, and produces a positive/productive timeline. |
|  | Developer | Architectural Requirements  Researches and implements designated frameworks. |
|  | Developer | Common Terms and Definitions  Researches and implements designated frameworks. |
|  | Developer | Design and Structural View: Class Diagram  Researches and implements designated frameworks. |
|  | Developer | Behavioral View: Sequence Diagrams  Researches and implements designated frameworks. |

**Common Terms and Definitions**

* CI/CD: Continuous Integration and Continuous Delivery.
* JUnit: Framework designed for writing and running unit tests for Java code.
* Jenkins: Automation server used to set up the CI/CD pipeline.
* Mutation Testing Framework: Used to determine the strength of code and tests by introducing various types of mutations for testing.
* Maven: A build tool used for Java projects.
* Pipeline: A set of automated processes to compile, build and deploy code.
* Plugin: Additional software that adds additional features to the host program.
* Spikes: Research on subjects to gain knowledge, reduce risks, or understand the requirements in greater detail.

**Project Scope**

Description:

A CI/CD pipeline is a method that helps developers have continuous integration and deployment for a development team’s application. Setting up a CI/CD pipeline ensures the team will have an easier time developing its product.

The CI/CD pipeline consists of several stages in the development process: build, test, release, deploy, and validate. The build stage includes the CI/CD pipeline taking the developer’s source code and being able to run it without flaws. The test stage usually consists of making unit tests of the source code to ensure the source code does not have bugs that will affect the system after deployment. This stage is our main concern for the project since we will be implementing mutation testing on the JUnit tests run in this stage. Automation in this stage with JUnit and the mutation test is key for a successful project. Once the test stage is complete, the developer should be able to release the application for deployment in the other stages already implemented in the CI/CD pipeline of their choice. The team will give a demonstration of their findings for client understanding.

Scope:

* To research and analyze different Java mutation testing frameworks and choose the best one to integrate into a CI/CD pipeline.
* To research how CI/CD pipelines work.
* To develop a plug-in that integrates into a CI/CD pipeline if no plug-in exists for the recommended framework.
* To make sure the Java mutation testing framework that would best integrate into a CI/CD pipeline will have continued support in the future.
* To develop a working program that provides software engineers with actionable information on the weak points in their unit tests.
* To have a framework that is automated through the CI/CD pipeline.

Milestones:

* Sprint 0: The team will research all Java Mutation Testing Frameworks. Produce a presentation of our research.
* Sprint 1: The team will choose a framework that integrates nicely into Jenkins by testing frameworks chosen in Sprint 0. Produce a presentation of our results and progress.
* Sprint 2: The team will demonstrate how already existing PiTest plugins, such as Autograding and PIT Mutation, are implemented into Jenkins. We will also discuss the Requirements and Design Documents.
* Sprint 3: The team will determine whether building a new plugin is necessary. Produce a presentation of our findings and progress.
* Sprint 4: Finish integrating/automating plugin into CI/CD pipeline. Document steps on how we got everything to work.
* Sprint 5: Make any necessary changes to customize or finish building mutation test plug-in. Provide a user manual and presentation.

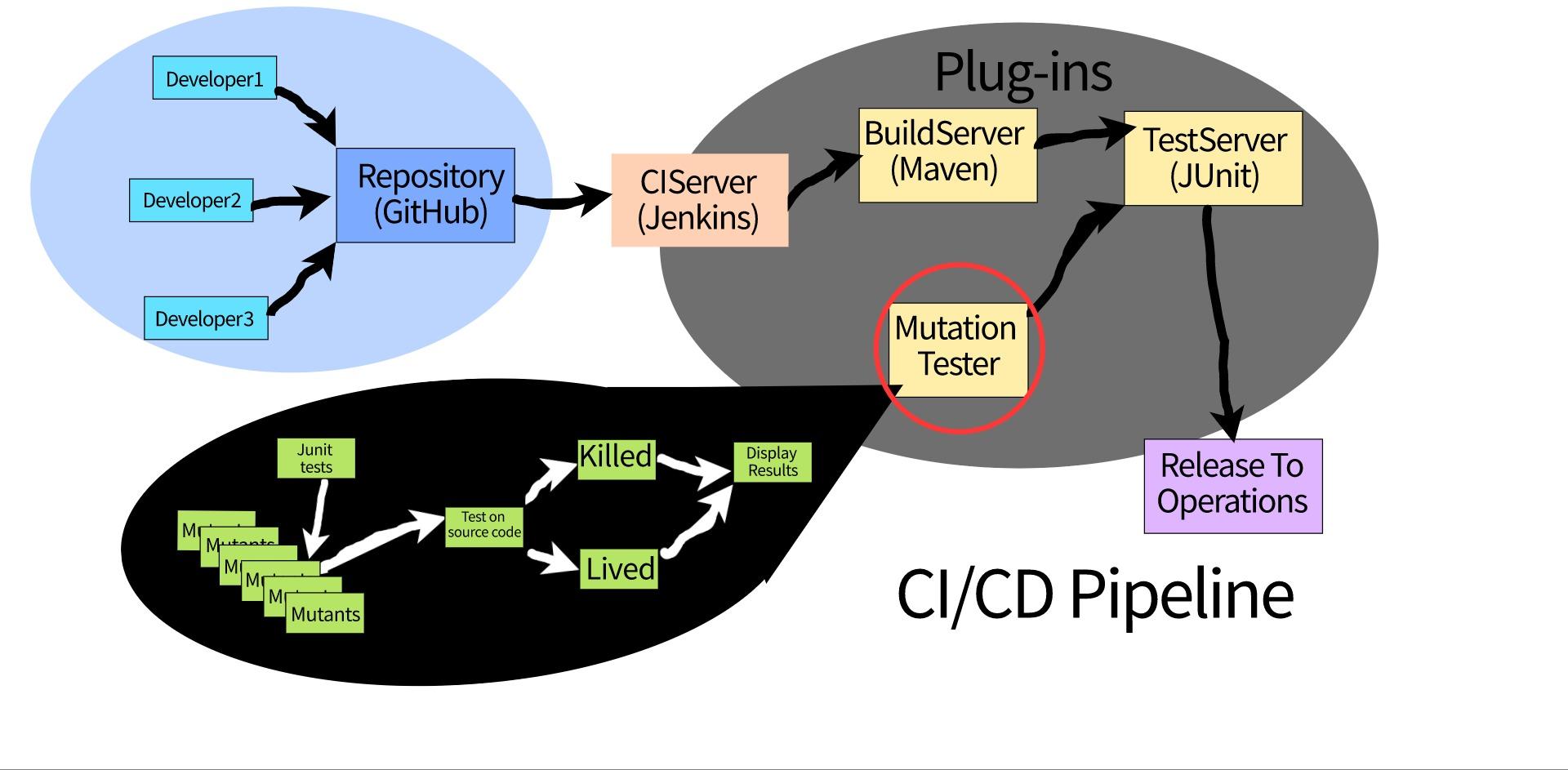
Constraints:

* Learning new things may result in improper testing of:
  + The framework, JUnit test, or maven configurations.
  + The pipeline, integration, and automation.
  + Pipeline script stage setup or placement.
* Redirecting value time due to:
  + Trial and error integrating plugins into Jenkins.
  + Getting Jenkins set up correctly with a CI/CD pipeline project.
  + Setting up an AWS server in an attempt to spend less time in future development.
  + Spikes needed to fill understanding for completing tasks.
  + Developing the supporting documentation. i.e., PID, Requirements Documents…

Deliverables:

Upon completion of the Java Mutation Testing Research Project, our client will be provided with an abundance of knowledge on how to automate PiTest in a CI/CD pipeline and a step-by-step guide for implementation in Jenkins. The developer will be able to use this guide to utilize the PiTest plugin to show results of their JUnit Tests in an easy and accessible way.

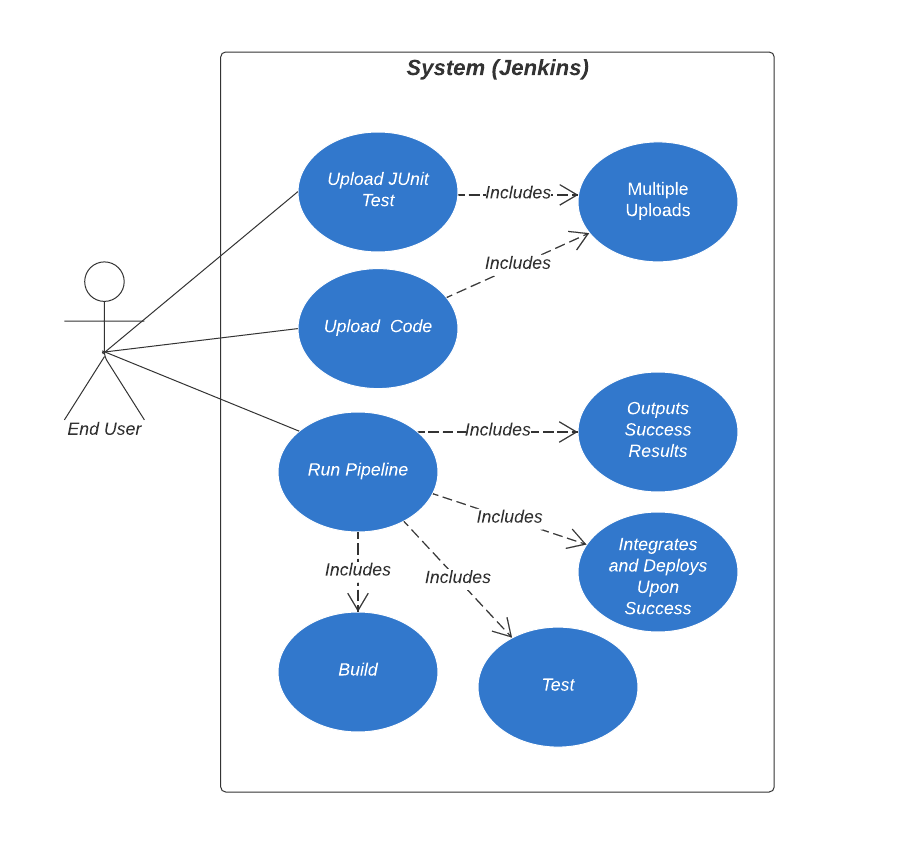
**Project Diagram**



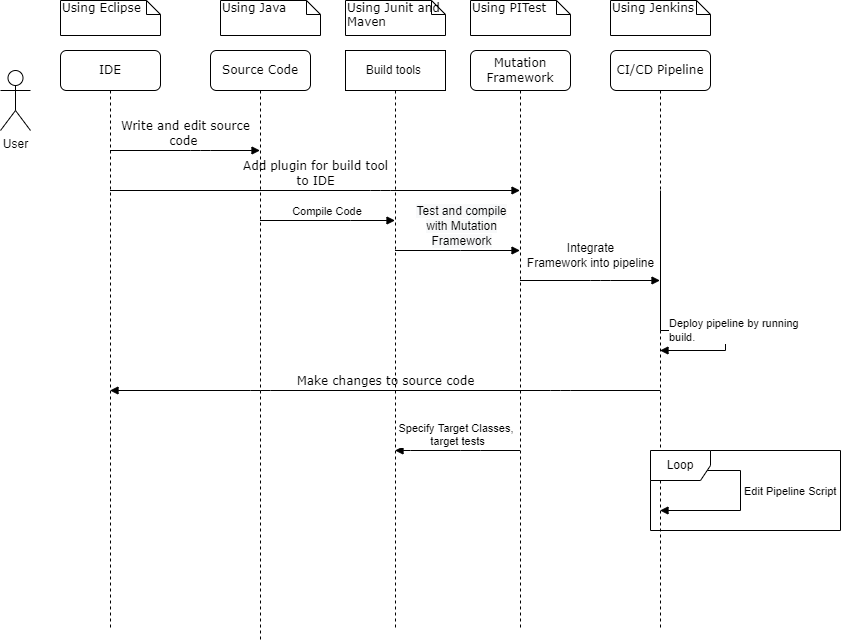
**Architectural Requirements**

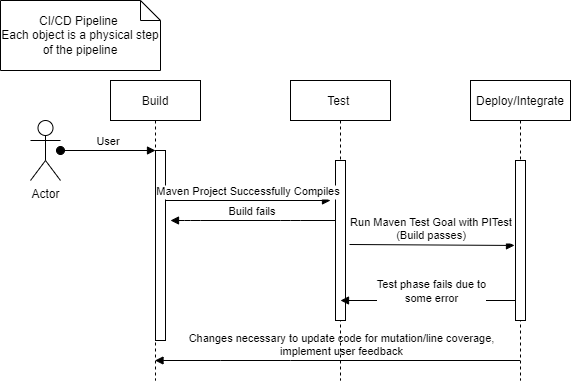
* The hardware requirements are specified in the CI/CD tool’s system requirements page. Webpages to download tools such as Jenkins will include documentation on what allotted RAM space, CPU, storage, and other requirements a computer needs for the tool to function.
* There are no input and output devices required to set up the automated mutation testing plugin in a CI/CD pipeline. All processes done within the pipeline are left to the user’s discretion. These processes can either be hosted on the user’s computer or through a server.
* A server is not necessary for the scope of this project. However, it is recommended to utilize a server for the necessary communication in a developmental team environment. Our team is using an Amazon web server for this purpose.
* Hosting a CI/CD tool, like Jenkins, can be done on a server or through the local network. The preference is up to the company/project.
* The interface consists of the Jenkins interface and whatever IED the user is using. In Eclipse we are integrating maven to run JUnit tests and PIT tests. Then in maven we can input the POM.xml file so Jenkins runs the JUnit test. We are trying to implement an automated CI/CD pipeline so the tests can be run over a certain period.

**Use Case Diagram View**



**Behavioral View: Sequence Diagrams**





**Design and Structural View: Class Diagram**

