**DevOps - Automation**

**Continuous Integration and Continuous Delivery (CI/CD)**

**A PROJECT REPORT**

*Submitted in partial fulfilment for the award of the degree*

*of*

**Master of Science**

***in***

**Information Technology**

*by*

**SANKET SURESH PETHKAR**

**(14MIN2879)**

*Under the guidance of*

**Prof. Sandeep Patil**

**VIT**

****



**School of Information Technology and Engineering**

June, 2018

****

**DECLARATION BY THE CANDIDATE**

I hereby declare that the thesis entitled **“DevOps – Automation (CI/CD Pipeline)”** submitted by me to Vellore Institute of Technology, Vellore, in partial fulfilment of the requirement for the award of the degree of **Master of Technology** in **Information Technology** is a record of bonafide project work carried out by me under the supervision of **Sandeep Patil**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**Place**:

**Date**: **Signature of the Candidate**

**School of Information Technology and Engineering**

****

**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled “**DevOps – Automation (CI/CD Pipeline)” by Sanket Suresh Pethkar (14MIN2879),** to Vellore Institute of Technology, Vellore, in partial fulfilment of the requirement for the award of the degree of **Master of Technology** in **Information Technology**, is a project bonafide work carried out by him/her under my supervision. The project fulfils the requirement as per the regulations of this Institute and in my opinion, meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

**Prof. Sandip Patil**

**Internal Supervisor**

**VIT**

**Internal Examiner(s) External Examiner(s)**

**Contents**

[1. Introduction vi](#_gjdgxs)

[1.1 Software Configuration and Release management vi](#_30j0zll)

[1.2 CI/CD vi](#_1fob9te)

[2. Issues in existing system vi](#_3znysh7)

[3. Proposed Solution vi](#_2et92p0)

[3.1. OBJECTIVES: vii](#_tyjcwt)

[4. Software and hardware specifications vii](#_3dy6vkm)

[5 Architecture Design viii](#_1t3h5sf)

[5.1 ARCHITECTURE DIAGRAM viii](#_4d34og8)

[5.1.1 UHNW viii](#_2s8eyo1)

[5.1.2 Conflicts Workflow ix](#_17dp8vu)

[5.2Use Case Diagrams x](#_3rdcrjn)

[5.2.1When to Use: Use Cases Diagrams x](#_26in1rg)

[5.3 Class Diagrams xi](#_lnxbz9)

[5.3.1 When to Use: Class Diagrams xii](#_35nkun2)

[5.4 Sequence diagrams: xii](#_1ksv4uv)

[6. Implementation xiv](#_44sinio)

[6.1STRATEGIC APPROACH TO SOFTWARE TESTING xv](#_2jxsxqh)

[6.2 SYSTEM TESTING xv](#_z337ya)

[6.2.1 Implementation and Evaluation xvi](#_3j2qqm3)

[6.2.2 Operation Evaluation: xvi](#_1y810tw)

[6.2.3 User Manager Assessment: xvi](#_4i7ojhp)

[6.2.4 Development Performance xvi](#_2xcytpi)

[6.3 System Implementation components include: xvi](#_1ci93xb)

[7. Test Plans xvii](#_3whwml4)

[7.1Test Objectives xvii](#_2bn6wsx)

[7.2 Extent of Tests xvii](#_qsh70q)

[8. Test design and Strategies xviii](#_3as4poj)

[8.1 General Test Strategy xviii](#_1pxezwc)

[8.2Integration Test Strategy xix](#_49x2ik5)

[8.2.1Test Case alignment with test phases xix](#_2p2csry)

[8.3Suspension and resumption xix](#_147n2zr)

[8.4 Automated Unit Test Suite xx](#_3o7alnk)

[9 Module Description xx](#_23ckvvd)

[9.1 Test Cases xx](#_ihv636)

[10. Reports xxvi](#_32hioqz)

[11. Abbreviations xxvi](#_1hmsyys)

**1. Introduction**

## 1.1Software Configuration and Release management

## Software configuration management (SCM) is a software engineering discipline consisting of standard processes and techniques often used by organizations to manage the changes introduced to its software products. SCM helps in identifying individual elements and configurations, tracking changes, and version selection, control, and baselining.

## Release management is the process of managing, planning, scheduling and controlling a software build through different stages and environments, including testing and deploying software releases.

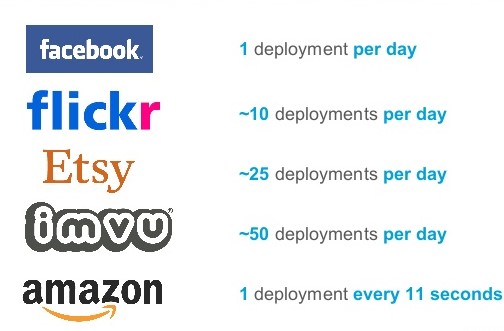
## 1.2CI/CD

**Continuous Delivery (CD)** is about automating Software Release Management (automated infrastructure provisioning, automated build, automated deploy and automated testing)

**Continuous integration (CI)** systems provide automation of the software build and validation process driven in a continuous way by running a configured sequence of operations every time a software change is checked into the source code management repository.

## 2. Issues in existing system

## In the current fast-moving IT world where everyone is moving to Agile, the number of deployments per unit time is increased so much that it is difficult to manage the infrastructure provisioning, build, deploy and testing manually. Manual work will need large number of resources, more time and there will be chance of errors and issues due to bulk work. To overcome all these challenges automation is required.

**Some Examples**:

**3. Proposed Solution**

**Implementing a CI/CD Pipeline for SDLC**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

**DevOps Practices**

The following are DevOps best practices:

* Continuous Integration
* Continuous Delivery
* Micro services
* Infrastructure as Code
* Monitoring and Logging
* Communication and Collaboration

## 3.1. OBJECTIVES:

Below are DevOps goals:

* Release on demand
* Eliminate technical debt and unplanned work
* Fail smart/fast/safe
* Look "outside-in"
* Measure feature value

## 4. Software and hardware specifications

**Hardware:**

Processor : Intel Core i5 or equivalent processor

RAM : Base machine – 8 GB or more

VM – 8 GB or more

ROM : Base machine – 1 TB or more

VM – 500 GB or more

**Software:**

Operating system : Base machine – Windows 7 or later

Front End Software : JAVA, HTML

Back End Software : SQL Server 2005

* Jenkins
* Nexus
* GitHub
* Ansible / Chef
* Docker
* MySQL, PostgreSQL
* Nginx, Jetty Server

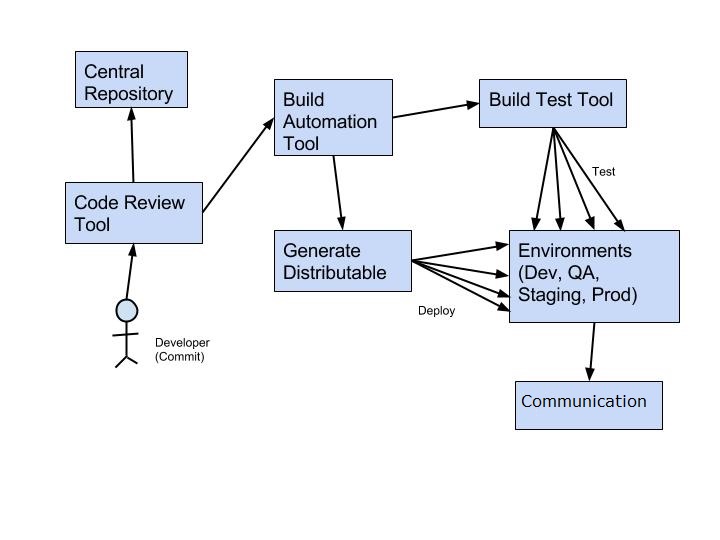
## 

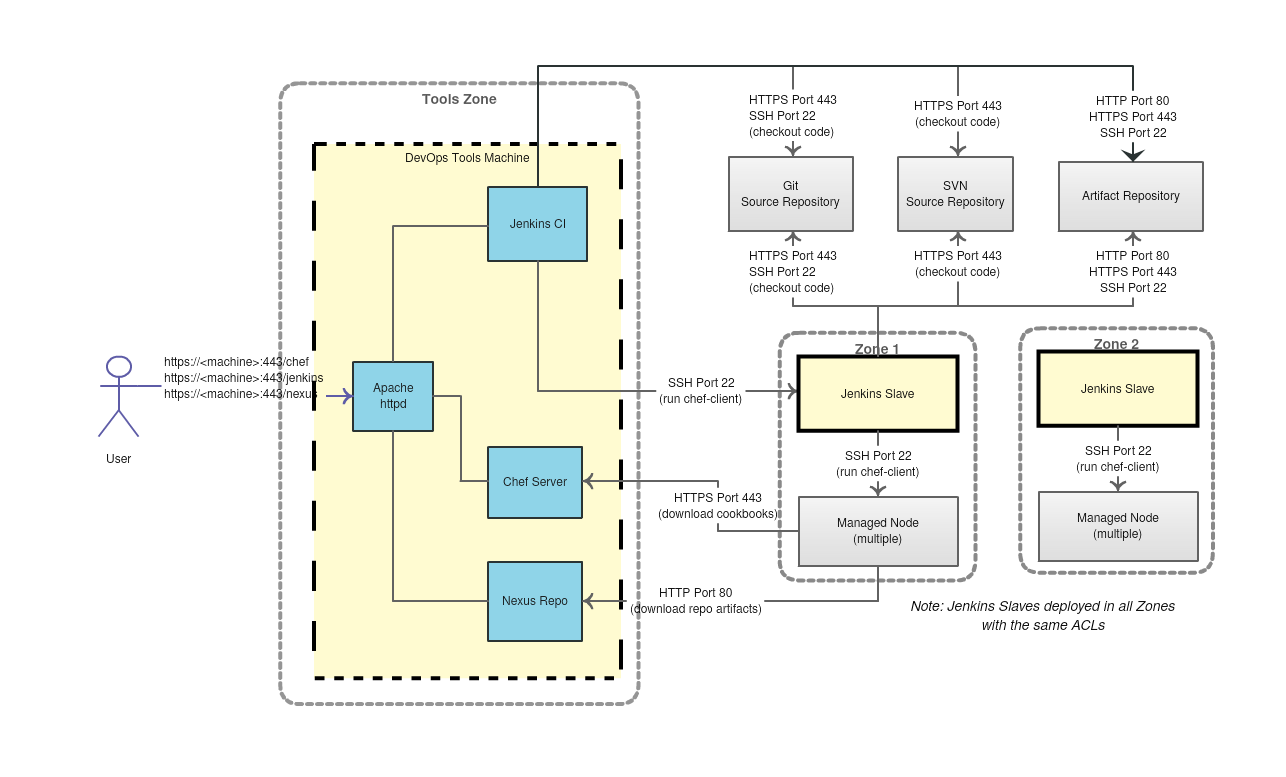
## 

## 5 Architecture Design

## 5.1 ARCHITECTURE DIAGRAM

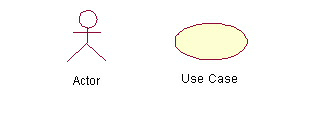
## C:\Users\SA292082\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CICDWorkflow.png5.1.1 CI/CD Pipeline Workflow

**5.1.2 DevOps Architecture**



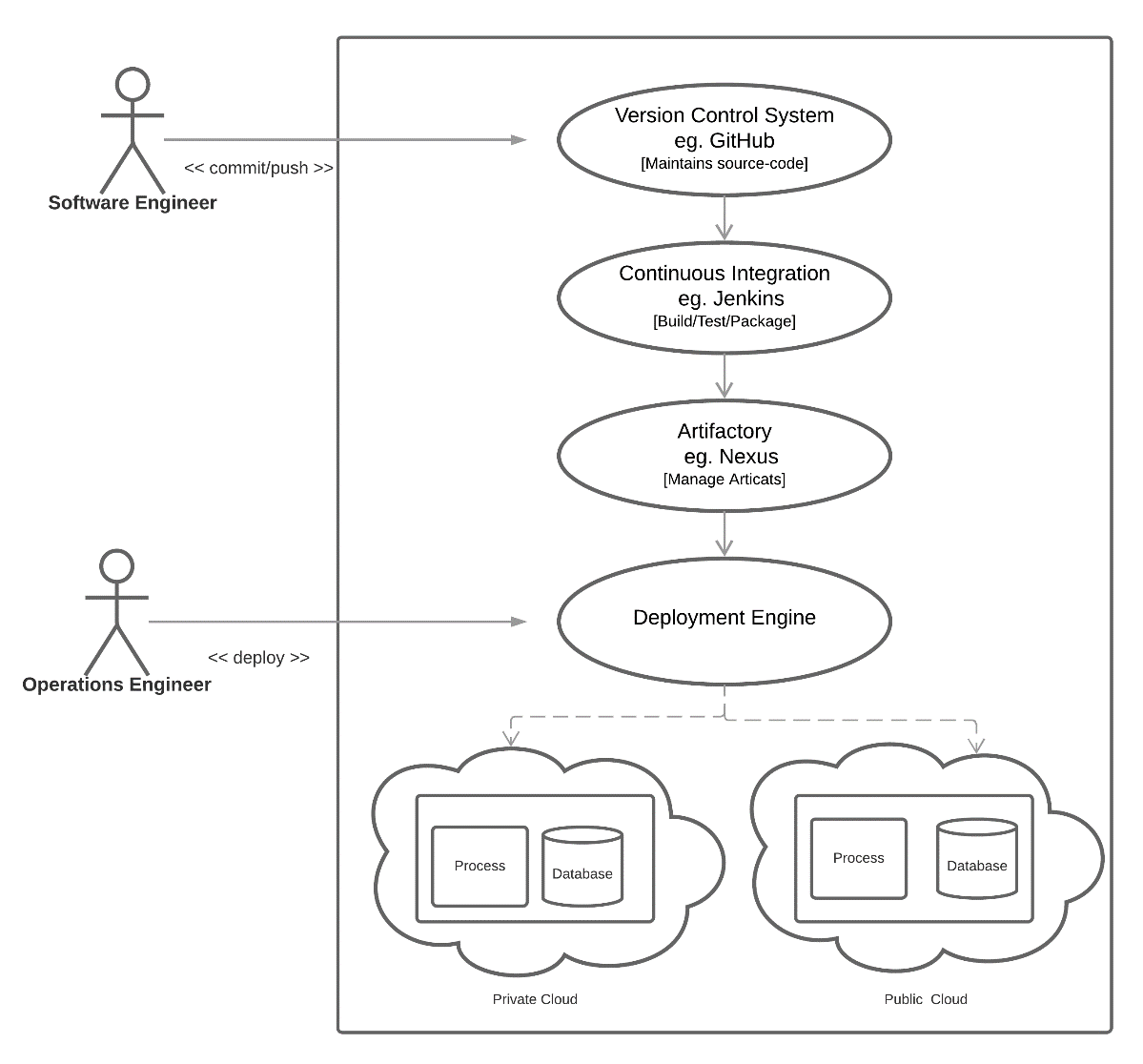
## 5.2 Use Case Diagrams

A use case is a set of scenarios that describing an interaction between a user and a system.  A use case diagram displays the relationship among actors and use cases.  The two main components of a use case diagram are use cases and actors.



## 5.2.1When to Use: Use Cases Diagrams

Use cases are used in almost every project.  They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

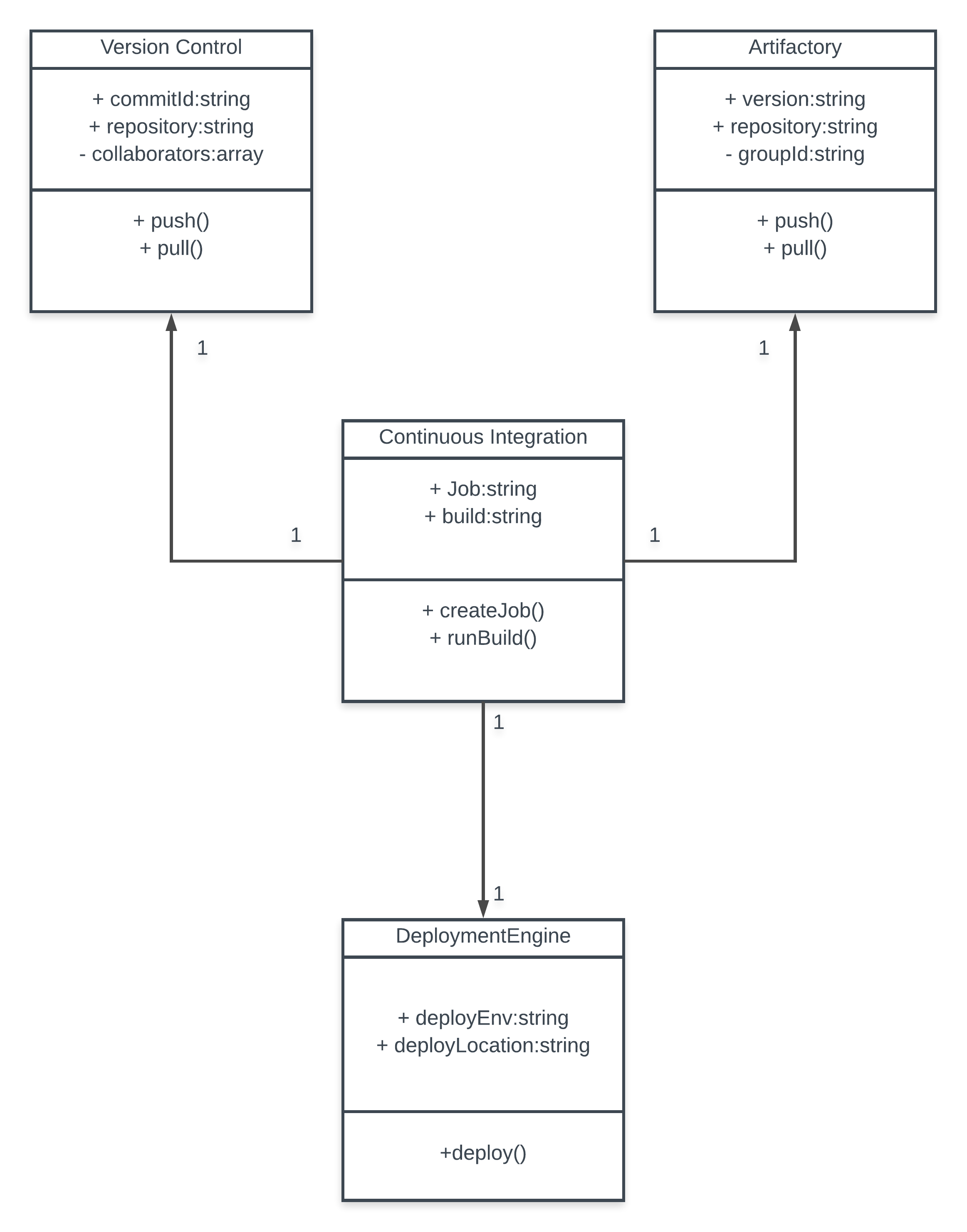


## 5.3 Class Diagrams

Class diagrams are widely used to describe the types of objects in a system and their relationships.  Class diagrams model class structure and contents using design elements such as classes, packages and objects.  Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design.  This example is only meant as an introduction to the UML and class diagrams.  If you would like to learn more see the Resources page for more detailed resources on UML.

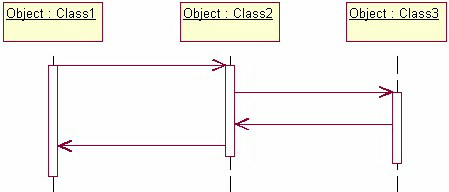
## 5.3.1 When to Use: Class Diagrams

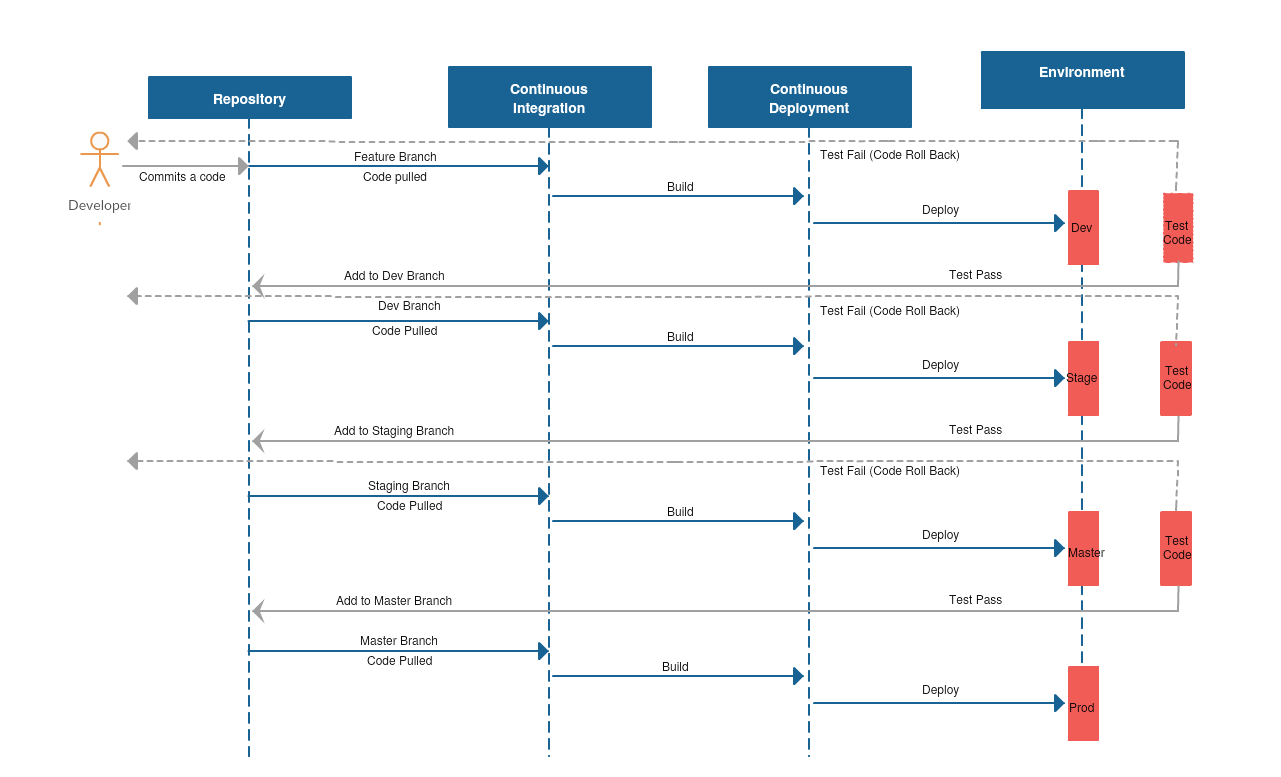
Class diagrams are used in nearly all Object-Oriented software designs. Use them to describe the Classes of the system and their relationships to each other.



## 5.4 Sequence diagrams:

Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass.  The diagrams are read left to right and descending.  The example below shows an object of class 1 start the behaviour by sending a message to an object of class 2.  Messages pass between the different objects until the object of class 1 receives the final message.





## 6. Implementation

## The increasing adoption of DevOps by most of the organizations across the globe clearly indicates its potential as a key enabler to achieving scale. Implementation of DevOps practices helps an organization deliver faster, better, high-quality and reliable software relying on the culture of cooperation and collaboration among all functions of an organization. It calls for fundamental cultural changes and modification of legacy programming practices.  Here are the core DevOps best practices that help an organization achieve the goals of effective communication and collaboration, smoother operations and bug-free code. Here are the ten key recommendations for successful DevOps implementation:

## 6.1 Continuous Integration

Continuous integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

We are using Jenkins as Continuous Integration tool which performs orchestration between all the tools like GitHub – SCM, Gradle – build tool, Nexus – Artifactory, etc. It controls and manages the flow of build pipeline.

## 6.2 Continuous Delivery

## Continuous delivery is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

## We are using Jenkins for deploying applications on the application servers. It will copy the application to apache tomcat server by unzipping the artifacts wherever required.

## 6.3 System Implementation components include:

**Personal Orientation:**

Introduce people to the new system and their relationship to the system

**Training:**

Give employees the tools and techniques to operate and use the system.

**Hardware Installation:**

Schedule for, prepare for, and then actually install new equipment.

**Procedure Writing:**

Develop procedure manual to follow in operating the new system.

**Testing:**

Ensure that the computer programs properly process the data.

**File Conversion:**

Load the information of the present files onto the new system files.

**Parallel Operation:**

Use the new system at the same time, as the old to make sure results are correct

## 7. Test Plans

The purpose of this document is to outline the test strategy and overall test approach for the AST Identity project. This includes test methodologies, traceability, and resources required, and estimated schedule.

## 7.1Test Objectives

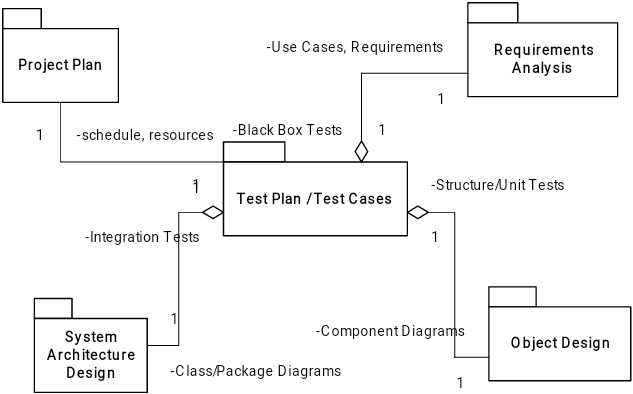
The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

## 7.2 Extent of Tests

The tests referenced herein are written to validate use cases, requirements (both functional and non-functional), system architecture, and object design. The structured tests for object design will be run first as the components of the system are developed. The structured tests to validate the system architecture will be run next as the system is integrated in bottom-up fashion during integration test.

A visualization of the relationships to the other documents can be seen in the diagram below.



## 8. Test design and Strategies

## 8.1 General Test Strategy

Unit testing and component testing will be performed on the components as they are developed. Test will be executed using test code in the form of either custom test tools or as an automated suite of tests run against the components in their individual sandboxes.

Integrations tests will be performed by both the component testers as well as the system testers. However, as the integration begins to include GUI level functionality, the tests being run will utilize significantly more manual testing and less automated testing.

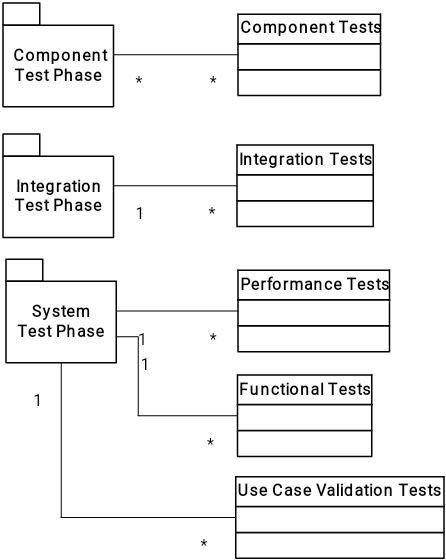
System test will require a new set of tools that can measure NFRS compliance, such as Jmeter (performance testing) or Quality Center (java source code analysis for security issues). Manual tests will start by validating functionality based on the requirements. Later stages of system test will include manual end-to-end tests to validate use cases.

## 8.2Integration Test Strategy

Because the components will be developed from the bottom-up and top-down, the test strategy will also align to the order of development of components. This will utilize a mostly bottom-up integration test approach, but will also involve the sandwich integration test approach.

Please review the system architecture overview and the subsystem architecture in architecture section

## 8.2.1Test Case alignment with test phases



# 8.3Suspension and resumption

This section specifies the criteria for suspending the testing on the test items associated with the plan. It also specifies the test activities that must be repeated when testing is resumed.

## 8.4 Automated Unit Test Suite

As components are being developed, unit tests will be developed to test the interfaces of the components and low-level unit tests will be developed to test the methods of the underlying classes in the components.

When the unit-test suite reports failures, testing will not occur on that build until the failures have been analyzed and resolved. Testing will resume on a build that passes the automated unit test suite.

## 9 Module Description

## 9.1 Test Cases

Scenario: To verify that Conflict officer is able to create new conflict record in Sugar.

|  |  |  |
| --- | --- | --- |
| Steps | Steps Description | Output |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Create Conflicts Review | New Conflicts form should be opened |
| Step 3 | When CO fills all the Details in Section 1: CS Division Transaction Type M&A Transaction Type Company Client Role Primary Project Secondary Project Lead Banker Secondary Banker Regions Countries Industries SpecialCategories Cross Border | All the details should be entered successfully |
| Step 4 | When CO fills all the details in Transaction Details section Status Transaction Notes Transaction Size Estimated Gross Fee Pitch Date Mandate Date Announced Date Completed Date Lost Date Dead Date NDA CA Status Engagement Letter Status Fairness Opinion Status Hostile/Friendly CS Role  Process | All the details should be entered successfully |
| Step 5 | When CO fills Clearance Notes and Code Name | Entry date should be populated |
| Step 6 | When user fills all the details in Clearance Instructions: Clearance Status Clearance Conditions Clearance Conditions Date  Clearance Instructions Banker Submission Notes | All the details should be entered successfully |
| Step 7 | When CO click on Save button | Then Conflict should be created successfully |

Scenario :To verify that CO if able to filter clients with Client name

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Client search New | Client Search Screen should be opened |
| Step 3 | When user selects Starts with radio button and select name from the filter and press enter key | All the clients should be loaded whose name starts with Name entered |
| Step 4 | When user selects contains radio button and select name from the filter and press enter key | All the clients should be loaded whose name contains Name entered |

Scenario : To verify that CO if able to filter clients with Client Code

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Client search New | Client Search Screen should be opened |
| Step 3 | When user selects Starts with radio button and select Client Code from the filter and press enter key | All the clients should be loaded whose Client Code starts with Client Code entered |
| Step 4 | When user selects contains radio button and select Client Code from the filter and press enter key | All the clients should be loaded whose Client Code contains Client Code entered |

Scenario : To verify that CO if able to filter clients with RIC Ticker

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Client search New | Client Search Screen should be opened |
| Step 3 | When user selects Starts with radio button and select RIC Ticker from the filter and press enter key | All the clients should be loaded whose RIC Ticker starts with RIC Ticker entered |
| Step 4 | When user selects contains radio button and select RIC Ticker from the filter and press enter key | All the clients should be loaded whose RIC Ticker contains RIC Ticker entered |

Scenario: TO verify that error message should be displayed while linking if project is already linked to a conflict

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Project Reconciliation Dashboard | PRD screen should be opened |
| Step 3 | Select and project and status and select Conflicts linked as Yes | Projects details should be opened |
| Step 4 | In Screen two click on Link button to link a conflict to a project | Verify the error message 'Project cannot be linked as it is already linked to some other conflict Record' |

Scenario : TO verify that error message should be displayed while creating if project is already linked to a conflict

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to Conflicts Review ->Project Reconciliation Dashboard | PRD screen should be opened |
| Step 3 | Select and project and status and select Conflicts linked as Yes | Projects details should be opened |
| Step 4 | In Screen two click on create button to create new conflict | Verify the error message 'Project cannot be created as it is already linked to some other conflict Record' |

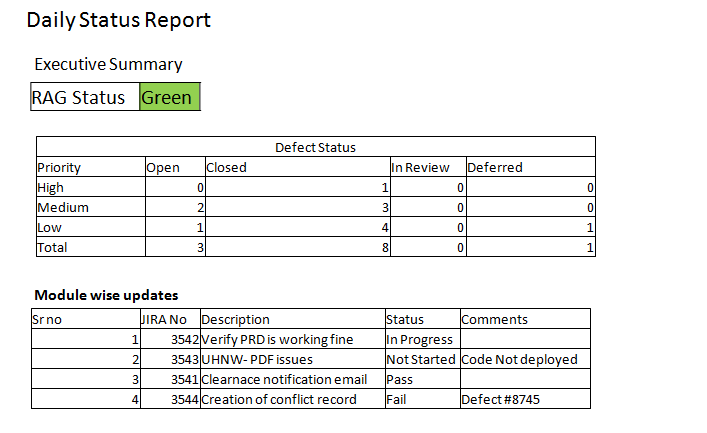
Scenario : To verify that UHNW requestor is able to create new draft records

|  |  |  |
| --- | --- | --- |
| Step 1 | Login to Sugar QA | Login is successful |
| Step 2 | Go to UHNW -> Create UHNW | New UHNW form should be opened |
| Step 3 | When Reqestor fills all the details in CS Contact Section CS Referral Contact1 Contact2 How was the Client obtained? Sales/Team Coverage | All the fields should be filled successfully |
| Step 4 | When Reqestor fills all the details in Customer Identification Information Entity or Individual? Spider ID Project Code Client Mnemonic First Name Last Name Middle Name Suffix Country of Birth DOB/Incorporation Date Government ID Type Government ID Identification Tax ID ype Tax ID Identification Tax ID no Address 1 Address2 City State Zip  Cuntry of Legal Domicile Email Address | All the fields should be filled successfully |
| Step 5 | When Reqestor fills all the details in Financial Information Section Employement Status Industry Years Employed Address1 address2 Address3 Source of wealth  Investable Assets Net Worth Valuation Comments | All the fields should be filled successfully |
| Step 6 | When UHNW requestor fills Risk as High or Low | Risk should be entered |
| Step 7 | When UHNW requestor fills all the Mandatory questions in EDD section | Alls the questions shouls be selected |
| Step 8 | When UHNW Requestor fills all the details in Financial Crime Risk Appetite Section | Details should be filled |
| Step 9 | Select the PEP staus: Yes No In Process | PEP status should be entered |
| Step 10 | After filling all the details clcik on save button | For should be create successfully and client ID no should be generated |

Scenario: To verify that Sr Client ID Analyst should be able to Approve, Reject or send the form for rework

|  |  |  |
| --- | --- | --- |
| Step1 | Login to Sugar QA as Sr Client ID Analyst | Login is successful |
| Step2 | Go to UHNW -> View UHNW | View Screen should be opened |
| Step3 | Select record with status 'Submitted to Client ID and click on edit | Form should be opened in edit mode |
| Step4 | In Review Approval Section Select : Approved Rework Reject | If Approved Form is sent to Cleint ID team. If Rewok then it is sent to Requestor team. If rejected then status is shown as rejected |

## 10. Reports



## 11. Abbreviations

CO – Conflict Officer

UHNW- Ultra High Net Worth

PRD- Project Reconciliation Dashboard