P&A | Test Strategy

Document	Products & Accounts - Test Strategy
Owner	@ Hemanshu Chauhan
Status	REVIEWED
Version	1.3

Note
*This test strategy **aligns** with:*

- ***Program Test Strategy & principles** defined at* https://ltmhedge.atlassian.net/wiki/spaces/EN/pages/575866337
- ***Program Testing Cadence** defined at* https://ltmhedge.atlassian.net/wiki/spaces/EN/pages/636387561/Testing++Cadence
- ***Program NFRs** defined at* https://ltmhedge.atlassian.net/wiki/spaces/EDOCS/pages/750455607/Non-Functional+Requirements

This is a Live document and subject to evolve with time.

1. Document Version & Review

1.1 Version Control

Date	Author	Role	Version	Comments
11 Nov 2019	@ Hemanshu Chauha	QE Lead - P&A	1.0	Initial Version
22 Jan 2020	@ Hemanshu Chauha	QE Lead - P&A	1.1	Updated Scope & multiple stages
12 Mar 2020	@ Hemanshu Chauha	QE Lead - P&A	1.2	Updated Overall structure & NFR
17 Mar 2020	@ Hemanshu Chauha	QE Lead - P&A	1.3	Updated Test Architecture & Stages

1.2 Document Review

Area	Reviewer	Reviewed
Product Owner	@ chris.wraith	☑
Engineering Lead	@ Vivek Upreti	☑
PM/ SM	@ ed.scott (Unlicensed)	☑
QE Lead	@ Hemanshu Chauhan	☑
BA Lead	@ harshita.pandey	☑

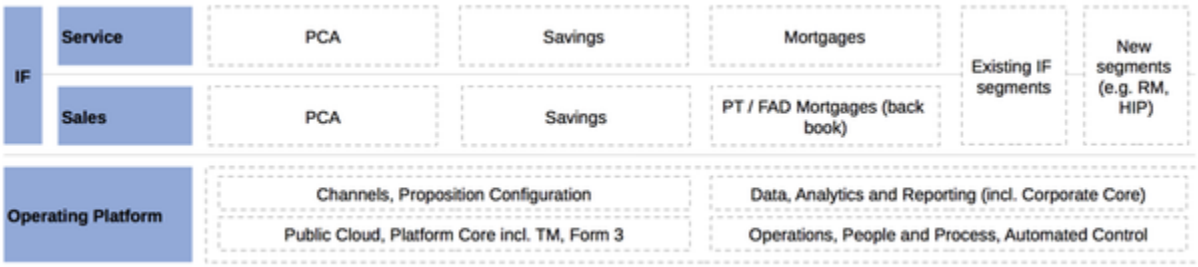
DL	@ gaurav.sehgal, @ anubhav.khanna, @ indra.sharma, @ Jo Shirley, @ Poulin, @ Neeraj Verma, @ Mohit Mair, @ Andrew James (Unlicensed), @ Nikhil Dass, @ sruthy.pramod, @ rishabh.govindraj, @ shivani.gupta9, LBGPAEndeavourPlus_PBS_TEAM_EMEA@publicissapient.com , @ Saurabh Sameer Saim, @ Bhargava Nallini, @ sachin.mehta, @ akos.fenemore, @ Sammy Voong, @ Huiqian.Lin (Unlicensed), @ vivek.singh1,

2. Overview

2.1 Overview of Endeavour+

The Endeavour+ scope (for 2020) will focus on build of a new strategic platform, Intelligent Finance (IF) back book migration and the launch of a digital front book that will have a deeper set of features than other challengers at launch.

Endeavour+ Scope; operating platform supporting front and back book for Intelligent Finance



2.2 Endeavour+ Scope

- Builds platform capable of supporting initial product scope of Intelligent Finance and foundations for subsequent options
- Builds operating structure for the new platform, including new control environment for public cloud / embedded automation
- Launches front book (aka. digital challenger MVP) and migrates back book for Intelligent Finance
- Creates new control framework commensurate with new processes / technology and people
- Intelligent Finance work touches prepare LBG, as migration proving activity is involved
- Builds insight and confidence in the Voyager business case *(TBC)*

Products & Accounts

Programme Test Cadences: Testing Cadence

2.3 Workstream scope

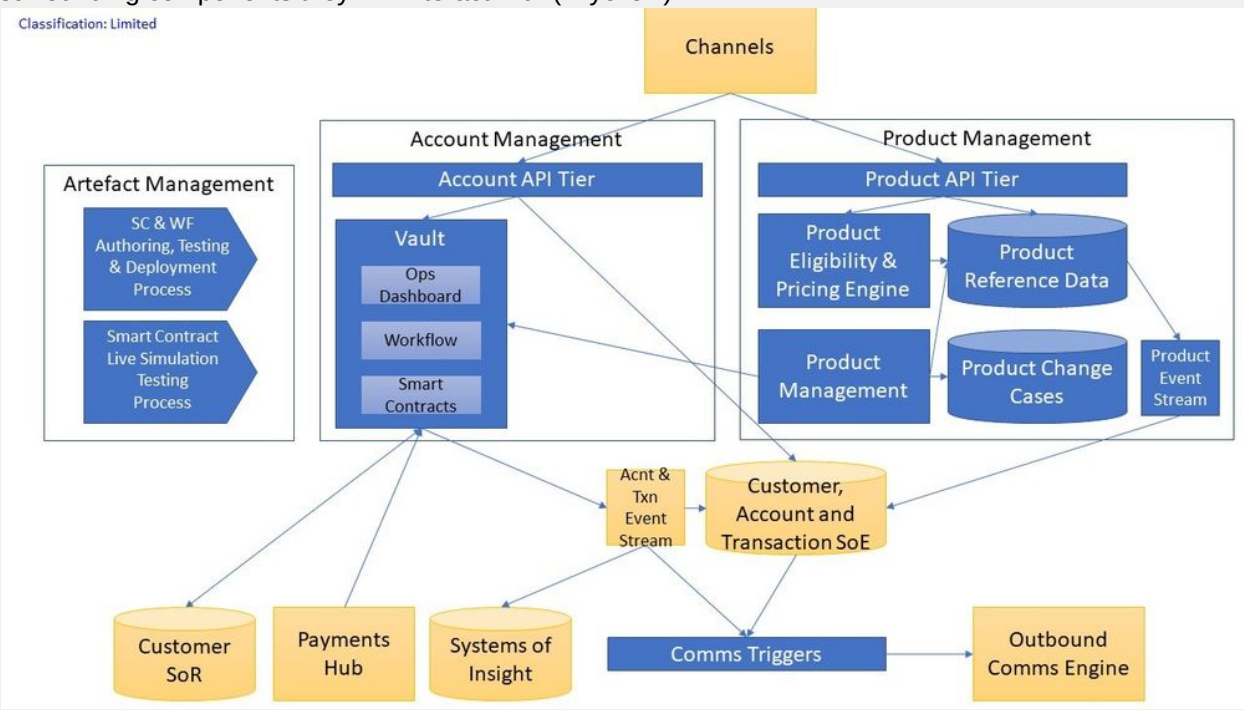
The project aims to deliver the Products & Accounts journey for IF customers holding DASA (Direct Access Savings Account Holder) account(s) on a live cloud/SaaS environment. Overall Product & Accounts scope and requirement are present here:

- **Product Management Requirements**

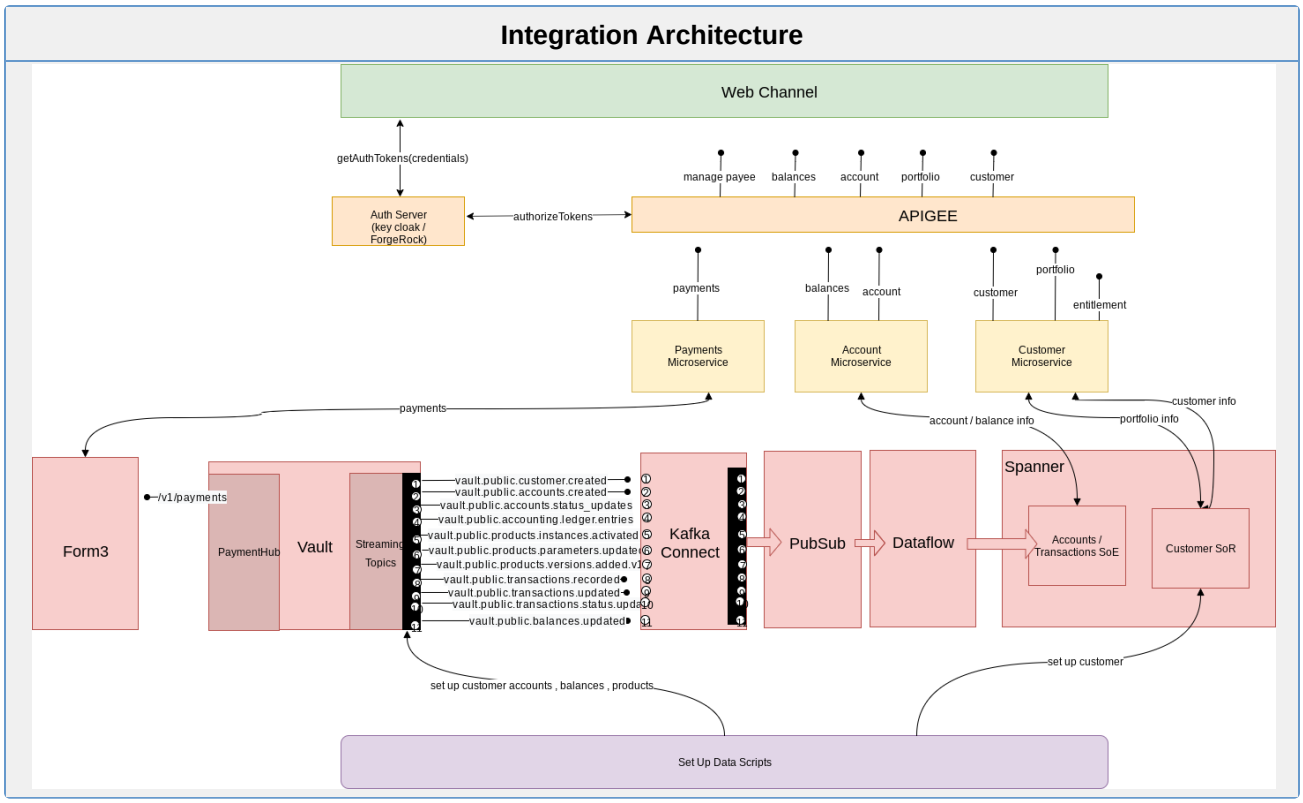
SCW Component Overview

Workstream Scope

The diagram below describes the high level components within this workstream (in blue) and the surrounding components they will interact with (in yellow):



Integration Architecture



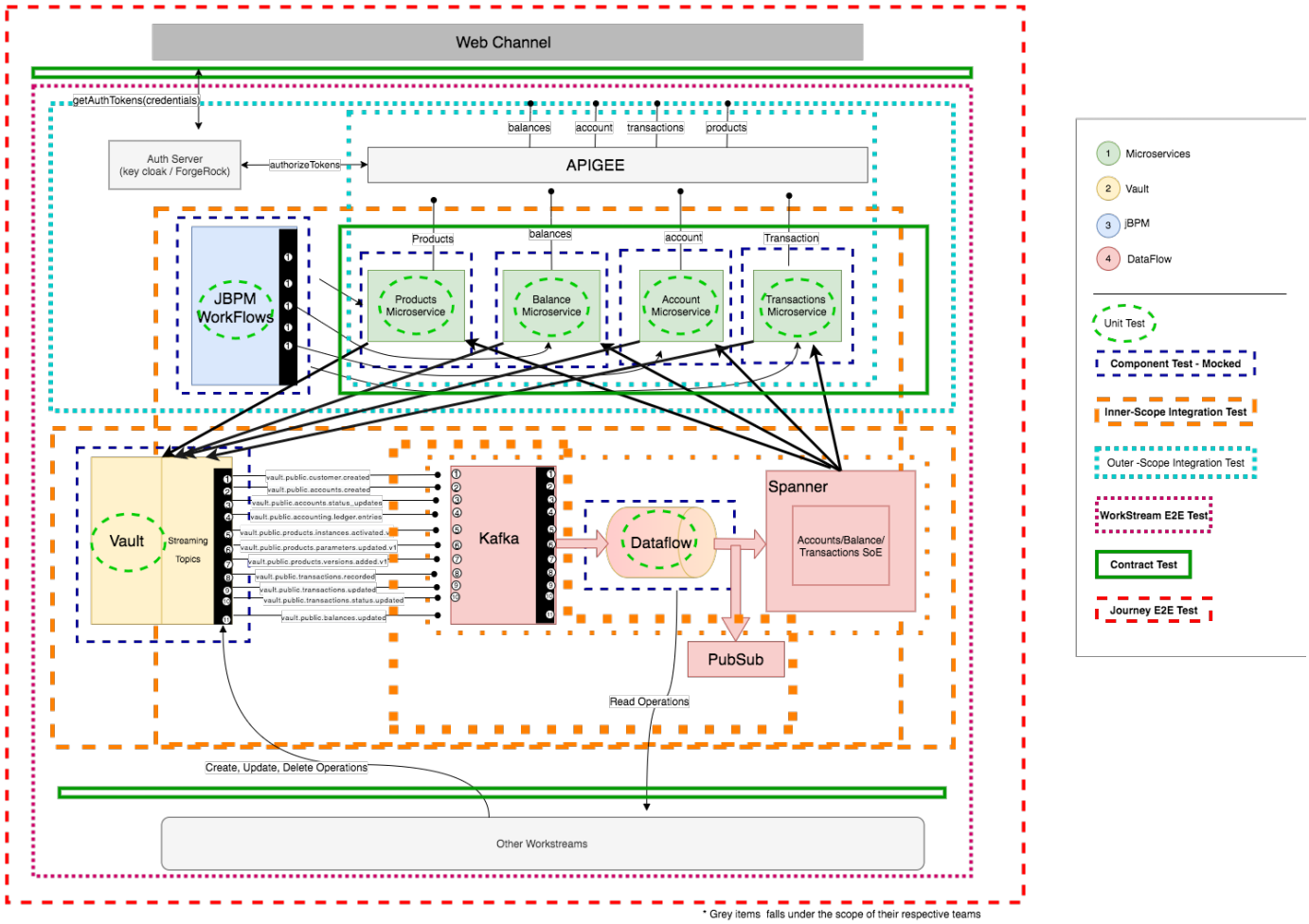
3. Purpose

The purpose of this Test Strategy is to define how the Smart contracts, Workflows and Microservices requirements would be tested in P&W workstream. Key Focus areas would be as below :-

- Test scope for Products & Accounts
- Test Architecture
- Test Approach
- Test Process
- Test Phases and Tools
- Entry & Exit Criteria
- Recommended Tools & Frameworks
- Test Data Management
- Test Management
- Defect Management
- Route to Live
- Reporting
- Key testing risks, issues, assumptions & dependencies
- Point of Contacts

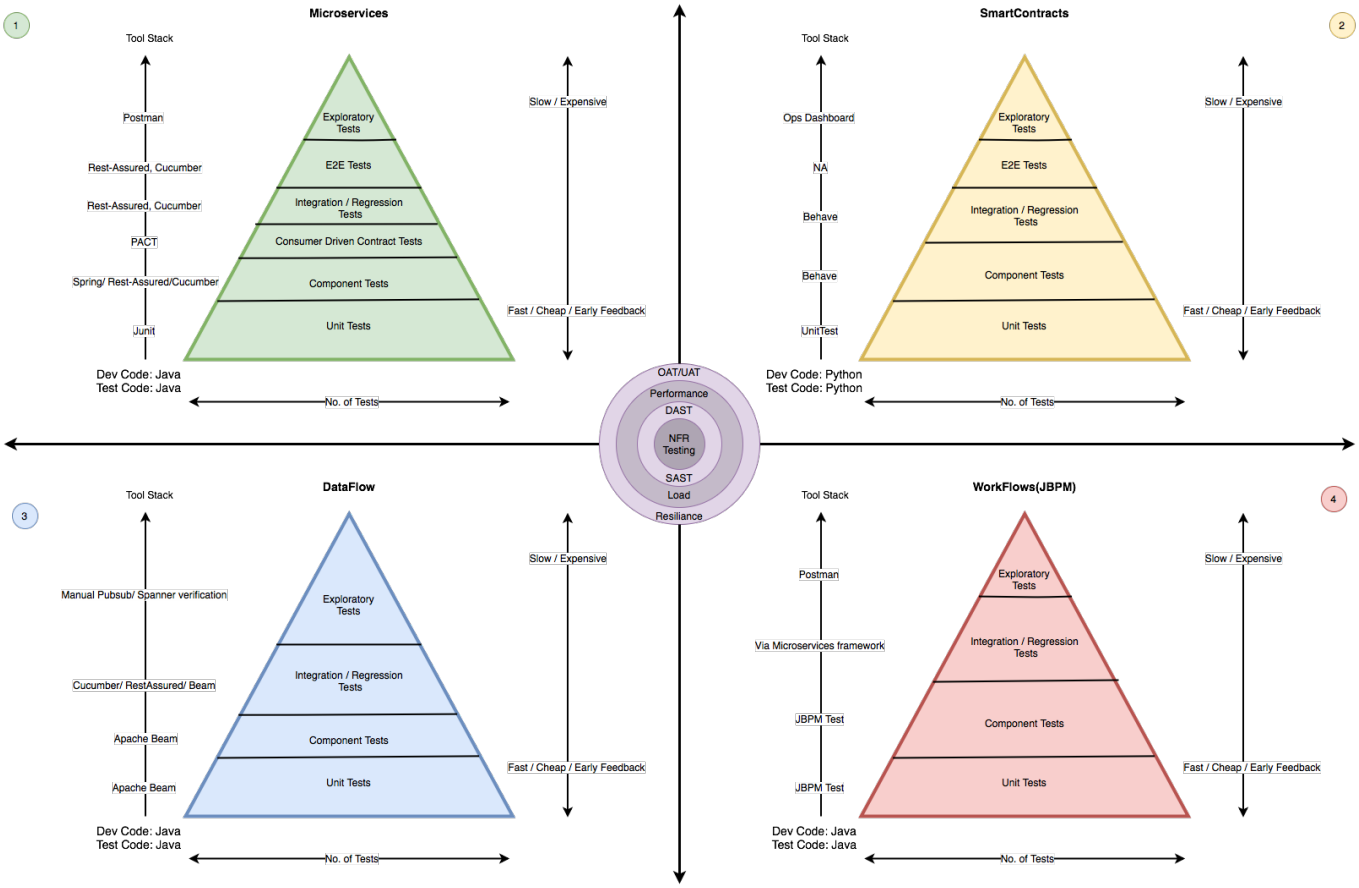
4. Test Architecture & Approach

4.1 Test Architecture



4.2. Test Approach

To Maintain Quality in "Products and Accounts" work-stream. We are following **"Test Pyramid"** approach.

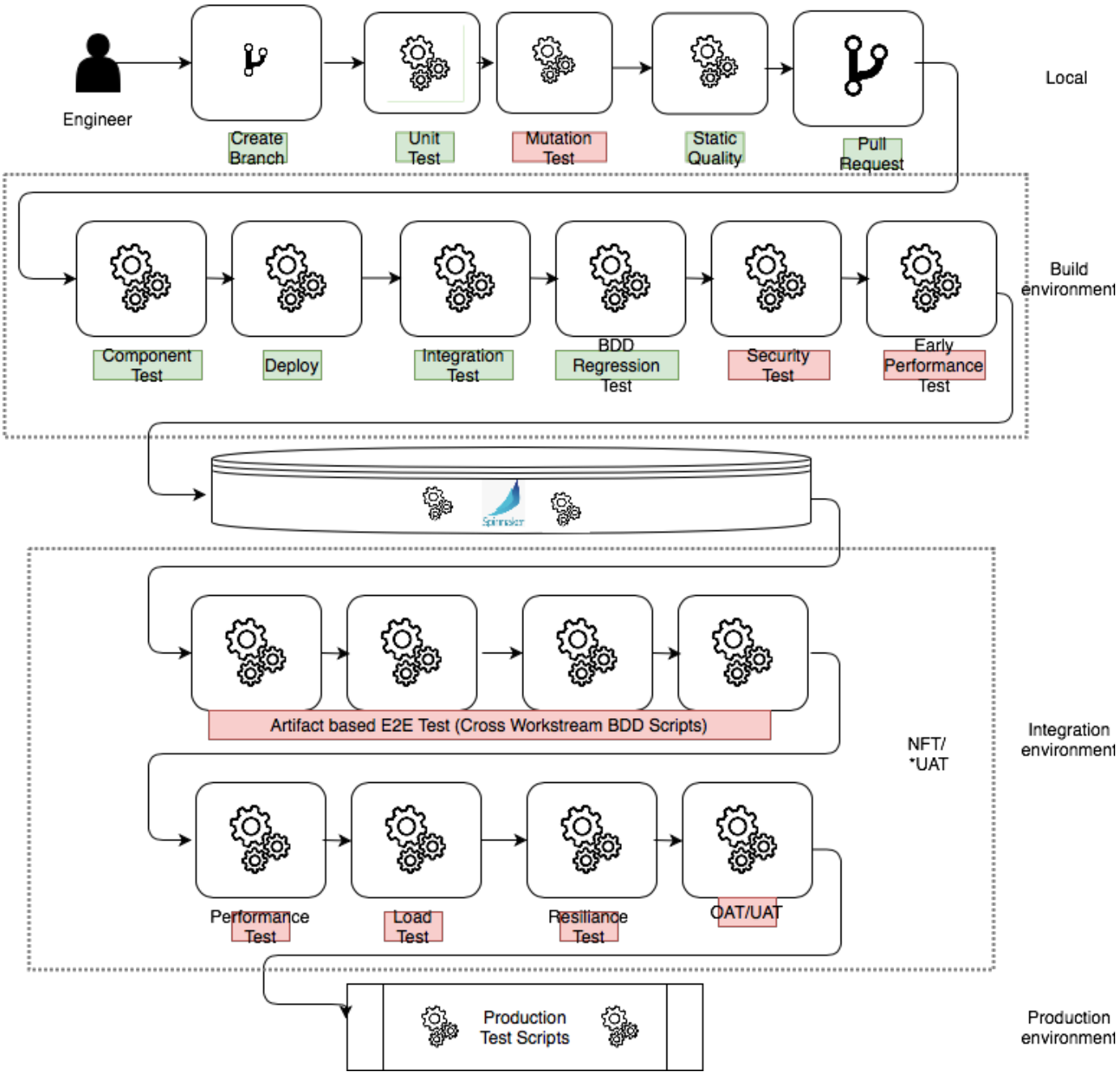


5. Test Process

This section defines the applicable test processes for environment proving that are aligned with the program level test approach. Apart from the programme level test processes, the below section explains the Backlog Refinement process to define the requirements as part of stories and align them within sprints.

- Stories that are to be covered as part of a sprint are picked up from the Product Backlog
- BA creates the Acceptance Criteria for every Story in JIRA
- Stories are refined and story pointed by the Team
- For every story, Automation Developer forks out a new branch from Automation Repo and writes Acceptance Scenarios in the Feature Files.
- BA reviews the Feature Files to ensure the functional coverage.
- Developer forks out a new branch from master Build feature Unit Test Component Test
- Automation Engineer build Integration/Regression scripts for the feature developed by developer and add then in pipeline
- Developer raises a patch set to push the code to Master, this is followed by code quality checks (like Sonar) and BDD suits written by automation engineer.
- Once, the Unit, Component & BDD pass percentage = 100% and the Dev Code is reviewed by the Architect, the code is pushed to Master.

6. Test Stages

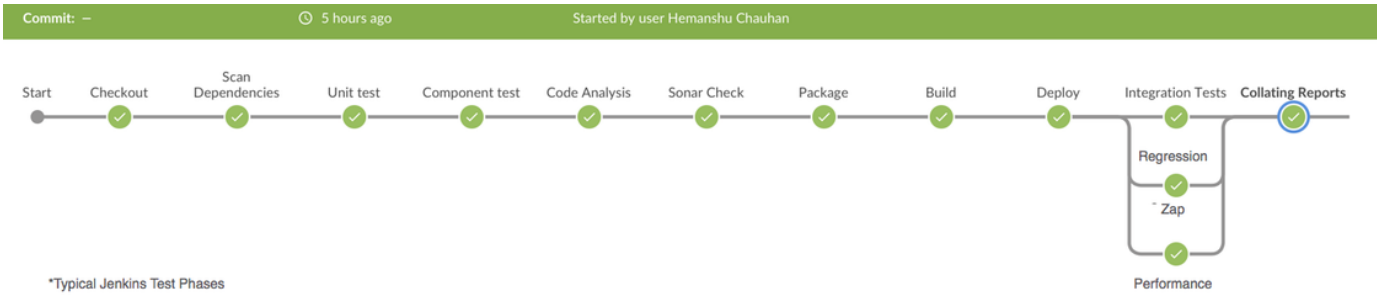


7. Test phases & Tools

7.1 Build Environment

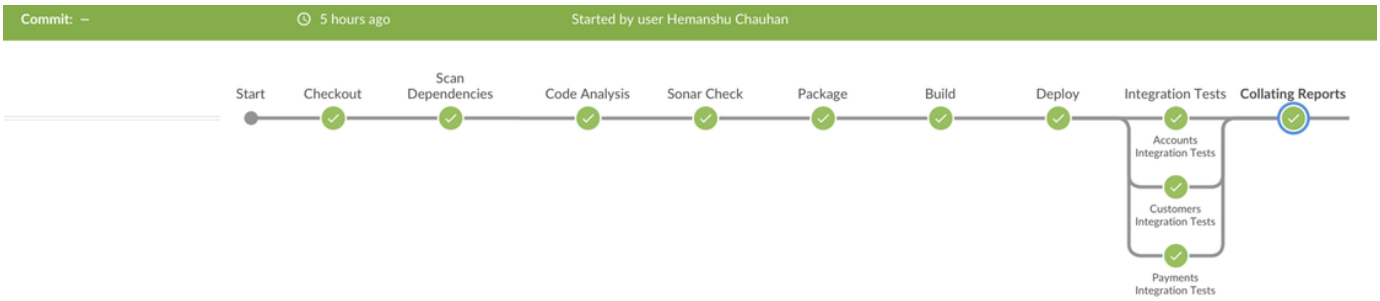
Functional Tests - Unit Test + *Mutation Test + Component Test + Inner-scope Integration Test + Regression Test

NFR Tests: Security + Performance + *Resilience Test + *Load Test



7.2 Integration & UAT Environment

Functional Tests - Outer-scope Integration Test & E2E



7.3 Production

*Privilege access approach to be discussed with governance & security team

Test Stages & Tools

Test Type	Component	Scope	Connectivity	AUT Tech Stack	Suggested Automation Tools /Frameworks	Code Repository	Ownership	Reviewer	Frequency of Execution	Jenkins Pass Threshold /Sign Off Criteria
Unit	Microservices	Classes, Functions etc. within a Microservice + Mutation Tests	In Isolation	Java	**Junit, Sprint-boot + Jumble/PIT(Mutation)**	**Development**	**Developer**	**Dev/ QE Peer**	Every Build (In-Sprint)	100%
	Smart Contracts	Classes, Functions etc. within a Smart Contract + Mutation Tests		Python	**Pytest + Mutpy (Mutation)**					
	JBPM Workflows	Classes, Functions etc. within a Workflow + Mutation Tests		jBPM /JAVA	**jBPM + Jumble/PIT (Mutation)**					
	DataFlow	Classes, Functions etc. within a DataFlow + Mutation Tests		Java	**Apache Beam + Jumble/PIT(Mutation)**					
Component	Microservices	Single Microservice	Mocks	Java	**Cucumber/ Sprint-boot/ RestAssured, Wiremock**	**Development**	**Developer**	**QE Peer / BA/ PO**	Every Build (In-Sprint)	100%
	Smart Contracts	Smart Contract (s)	Simulators	Python	**Behave**					

	JBPM Workflows	Workflow(s) with/without Smart Contract (s	APIs	jBPM /JAVA	**Cucumber, RestAssured, jBPM**					
	DataFlow	Individual Dataflow pipeline	Mutators	Java	**Apache Beam**					
Integration (Inner Scope + Outer Scope)	Microservices	Multiple intra & inter workstream Microservices interacting with each other	Real	Java	**Cucumber, Rest-Assured**	**QE**	**QE**	**QE Peer/ BA/ PO**	Every Sprint (Inner Scope) + Release Basis (Outer Scope)	100% (Exceptions if any approved by PO)
	Smart Contracts	Smart Contract deployed to Vault	Vault	Python	**Behave, requests**					
	JBPM WorkFlows	Workflow(s) connected with APIs	APIs & Vault	Java, jBPM	**jBPM, Cucumber, RestAssured**					
	DataFlow	Integrated DataFlow Pipeline	Spanner, PubSub, SOR, SOE, SOI	Java	**Apache Beam, RestAssured, Cucumber**					
End to End	• Microservices • Smart Contracts • JBPM Workflows • Dataflow • **Data Migration**	Single/Multiple Microservices + SOE	Spanner, Pubsub, SOE, SOR, SOI	Java	**Cucumber, Rest-Assured, DataFlow Utility (Apache Beam), Spanner Utility, Pubsub Utility, Data Reconciliation utility**	**QE**	**QE**	**QE Peer /BA/PO**	Release Basis	100% (Exceptions if any approved by PO)
	• Microservices • Smart Contracts • JBPM Workflows • Dataflow • **Data Migration**	Apigee + Single/Multiple Microservices + SOE	Spanner, Pubsub, SOE, SOR, SOI	Java	**Cucumber, Rest-Assured, DataFlow Utility (Apache Beam), Spanner Utility, Pubsub Utility, Data Reconciliation utility**					
					Non - Functional					
Performance & Load Testing	• Microservices • Smart Contracts • Workflows • Dataflow	Performance Baseline Testing of Java Microservices	Real	Java/ python	**Gatling** *InfluxDB, Graphite, Grafana*	Development /QE	QE	QE Peer	Nightly Scheduled & Release Basis	Workstream Baseline
Security Testing (DAST /SAST)	• Microservices • Smart Contracts • Workflows • Dataflow	Dynamic & Static Code Scans	Real	Application Code	**SonarQube, Zap, Veracode**	Development /QE	QE/Dev	Dev/QE Peer	Every Build (In-Sprint)	100%
INT / UAT	• Microservices • Smart Contracts • Workflows • Dataflow	User Acceptance Flows	Real	Java	**Automated Integration Packs**	QE	QE	PO	Release Basis	Workstream Baseline
Penetration Testing	• Microservices • Smart Contracts • Workflows • Dataflow	Security NFRs	Real	Java	TBD	NA	TBD	TBD	Release Basis	TBD

8. Scope

8.1 Test Scope

Products and Accounts work-stream is majorly doing development in 4 main areas:

Components & Stages	Functional					Non-Functional						
	Unit	Component	Integration	E2E	CDC	Security - DAST	Security - SAST	Performance	Load	Operational	Resilience	Penetration
Microservices	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	Security Team
Smart Contracts	In-Scope	In-Scope	In-Scope	In-Scope	NA	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	Security Team
JBPM Workflows	In-Scope	In-Scope	In-Scope	In-Scope	NA	In-Scope	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	In-Scope (WIP)	Security Team
TM Workflows	In-Scope	In-Scope	In-Scope	In-Scope	NA	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	Security Team
DataFlow	In-Scope	In-Scope	In-Scope	In-Scope	NA	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	Security Team
Spanner	NA	NA	In-Scope	In-Scope	NA	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	Security Team
Pubsub	NA	NA	In-Scope	In-Scope	NA	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	Security Team
Data Migration	In-Scope	In-Scope	In-Scope	In-Scope	NA	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	In-Scope	Security Team

8.2 Brand Scope

This section defines which brands are in the scope of this project.

Brand	In Scope
Intelligent Finance (IF)	Yes
St. James Place Bank (SJPB)	Yes
Lloyds, BOS, Halifax	No

8.3 Product Scope

Following Account Categories are in-scope for steel thread.

Product Type
Savings Accounts
Current Accounts
Mortgages Accounts

Detailed List: IF & SJPB Product Information

9. Entry and Exit Criteria

The entry and exit criteria are based on the following sub sections which define the process to deliver functional stories:

Definition of Ready (DOR)	Definition of Done (DOD)

Engineering	**Engineering**
Stories are ready for sprint when … 1. Requirements are clear (to the team) 2. Acceptance Criteria are defined and established as 3. They have been refined 4. Inter-team dependencies have been identified and managed 5. Have been sized Where applicable: 1. NFRs are identified 2. Environments are available 3. Test data is prepared 4. Architectural decisions have been made	Stories are done when the … 1. Code has been implemented 2. Acceptance Criteria (ACs) have been met 3. CI\CD has passed successfully (All Unit Tests, Security Tests & Component Tests are passing) 4. Code review has been approved and code has been merged to Master 5. It has been deployed to highest available environment 6. Relevant documentation updates completed (Support manual, user guide, high level design) - As applicable

10. Test Management

10.1 Supporting Documentation

All important test documents will be maintained in confluence & Jira.

10.2 Defect Management

All defects detected during testing will be logged and reported using the Defect management System in JIRA. When a suspect defect occurs during test execution the tester may attempt to repeat the test and diagnose the problem. If this takes more than half an hour without a result, the tester will simply log the issue in Jira keeping Test Lead, Engineering Lead and PO in loop.

The priority of a defect is the l indicates the importance or urgency of fixing a defect. Its values would be the same if the error occurred in a test or in a production / live environment. Priority ranges from the following:

Priority	**Impact of Defect to go live**
Highest	This problem will block progress.
High	Serious problem that could block progress.
Medium	Minor problem or easily worked around.
Low	Minor problem or easily worked around.
Lowest	Trivial problem with little or no impact on progress.

10.3 Test Data Management *(WIP)**

SCW team will create their own test data in Vault and Spanner. Test Customers

10.4 Test Packs

Products & Accounts work stream will create test packs around Microservices, Smart contract, Workflows & DataFlow. These test packs will be in BDD/Cucumber format and will be executed on the basis of tags and features.

For example: *@regression* tag will be used in all feature files for regression execution; while *@sanity* tag will be used for sanity execution of some specific scenarios.

```
@api @regression @accounts @getaccount
Feature: ACCOUNT DETAILS | Get Account details using Account ID

########################################## Positive Scenarios ########################################

  @scenario1 @getaccount @positive
  Scenario Outline: User should be able to get account details by account id using "getAccount API"
    Given I am a JSON API consumer
    And I am executing test "GA1"
    And I provide the header "x-lbg-brand" with a value of "<Brand>"
    When I request GET "<Endpoint>"
    Then I should get a status code of 200
    And the response value of "Data.Account[0].AccountId" should equal "8e79edcf-01e7-2fa0-890e-243baa82c7fb"
    And the response value of "Data.Account[0].Account[0].schemeName" should equal "UK.OBIE.SortCodeAccountNumber"
    And the response should contain the following elements
      | Data.Account[0].AccountId              |
```

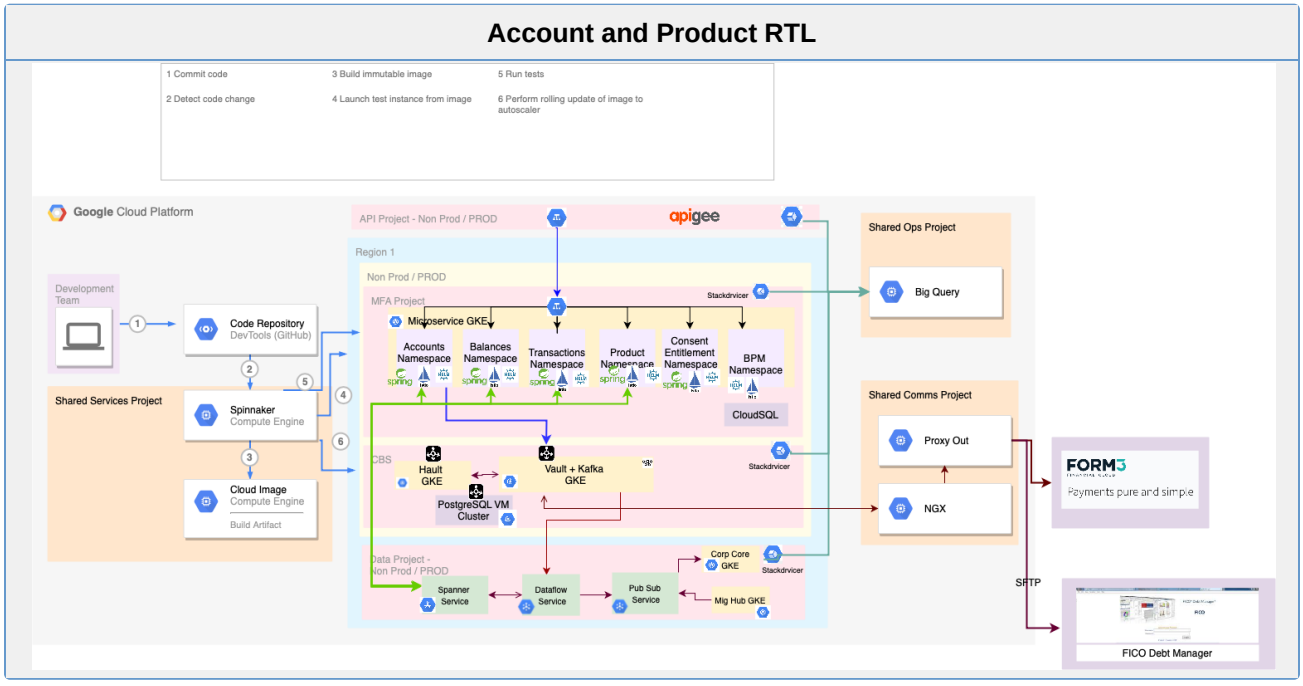
11. Monitoring & Control*(TBD)*

First level of application stability and monitoring will be done via Jenkins Jobs through scheduled BDD or automation suite execution. These Jenkins Jobs will generate recurring email notifications and Allure reports will be available for detailed view in Jenkins

11. Release Management - *(WIP)*

11.1 RTL Plan

Account and Product RTL



11.2 Version Controls - *(tbc)*

11.2 End Of Test Report (EoTR) - (link to be added)

12. Test Environments

Test Environment details are present in RTL process: Account and Product RTL

13. Reporting

13.1 Single End-Of-Test Report (EoTR) will be created for every release(code push beyond INT environment) with all Functional, NFT, UAT and other Sign Off's.

EoTR covers the following -

1. Release Scope
2. Pass % (should be 90% or more)
3. Test Execution Results
4. Point Release (if any)
5. Environment Sign off
6. Defect(s) Raised
7. Any open defect

Every open defect should be signed off from PO.

13.2 Allure/Cucumber Reporting

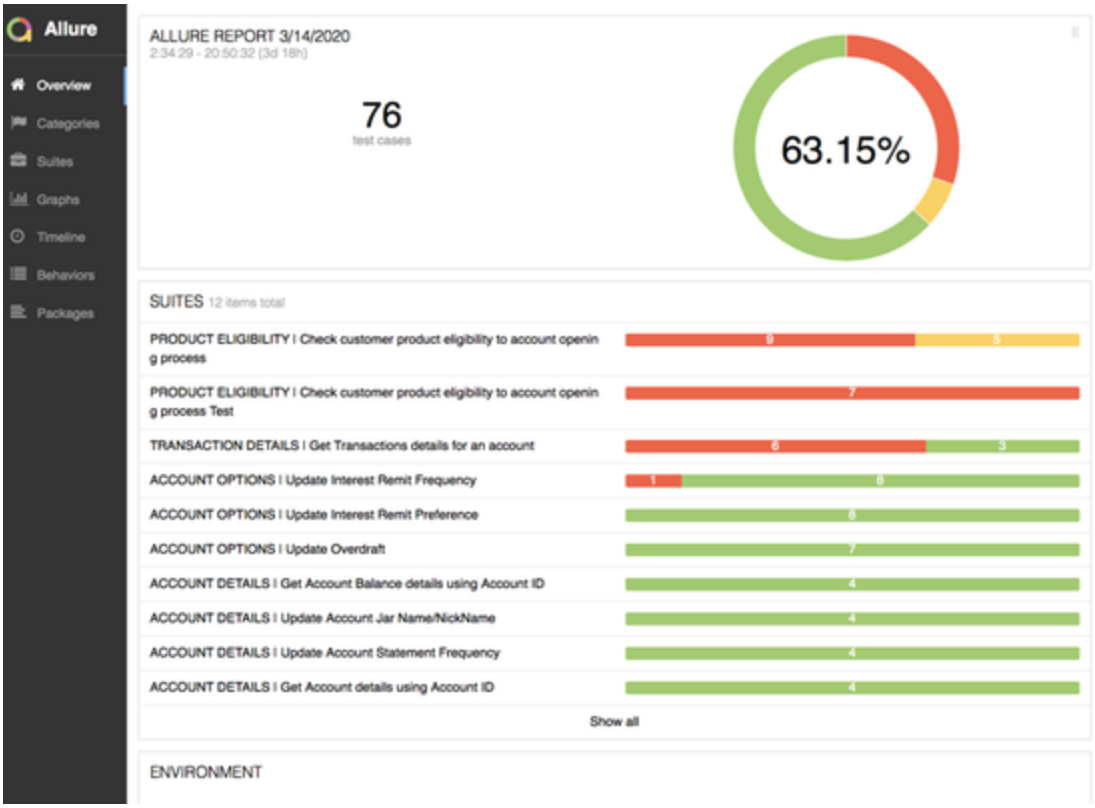
BDD Execution Reports (HTML), depicting the below data will be present in Jenkins CI/CD pipeline for each feature and repository:

Feature View - Depicts the following:

- PIE Charts showing the Features Pass/Fail %age & Count
- PIE Charts showing Scenario Pass/Fail %age & Count
- Features - Pass/Fail/Total Count
- Scenarios - Pass/Fail/Total Count
- Steps - Pass/Fail/Skipped/Pending/Undefined/Total Count
- Duration for every Feature/Step
- Feature/Scenario/Step Description
- Error reason (in case of failure)
- UI Snapshot (in case of UI Scenario)

Tag View - Depicts the following:

- Tag Statistics - Bar representation of Tags Execution
- Scenarios - Pass/Failed/Total Count for each Tag
- Steps - Pass/Failed/Skipped/Pending/Undefined/Total Count for each Tag



13.3 Speedy Reporting

Speedy Dashboard contains abstract Report of all the Quality Gates in the DevOps Pileline.

Report includes:

- BDD Pass Trends
- Voilation Trends
- Unit Test Voilation Trends

14. Risks, Issues, Assumptions & Dependencies

14.1 Assumptions

Where-ever possible, team will use Mocks/Stubs to remove external dependencies

14.2 Dependencies/Risks

Below are few dependencies which can impact test cycles and execution:

- Cross workstream dependencies & releases
- *tbc*

15. Point of Contacts

Mentioned below are the POCs for different items in Products & Account Workstream:

Name	Role	Email ID
Chris Wrath	Product Owner	chris.wraith@lloydsbanking.com
Vivek Upreti	Engineering Lead	vivek.upreti@publicissapient.com
Ed Scott	Scrum Master	ed.scott@publicissapient.com

Neeraj Verma	Tech Lead	neeraj.verma@publicissapient.com
Hemanshu Chauhan	Lead Quality Engineer	hemanshu.chauhan@publicissapient.com
Poulin Michael	Sr. Quality Engineer	poulin.michael@publicissapient.com