CARD PAYMENT FROM MERCHANT

Test Plan

Document Change History

Version Number	Date	Contributor	Description
V1.0	18/05/2021	Rashmi S	Document Created
V2.0	21/05/2021	Rashmi S	Updated the test feature

Table of Contents

1 Introduction

This test approach document describes the appropriate strategies, process, workflows and methodologies used to plan, organize, execute and manage testing of software projects.

1.1 Scope

The scope of the project is to enable the Card payment for a customer via Merchant by making valid security checks are in place before processing the request and sending back the response to the payment system

1.1.1 In Scope

The *Card Payment from Merchant Test Plan* defines the unit, integration, system, regression, and Client Acceptance testing approach. The test scope includes the following:

- Testing of all functional, application performance, security and use cases requirements listed in the *Test Case* document.
- Quality requirements.
- End-to-end testing and testing of interfaces of all systems that interact with the payment processing system.

1.1.2 Out of Scope

The following are considered out of scope for Card Payment from Merchant system Test Plan and testing scope:

- Functional requirements testing for systems outside such as the DW, Fraud Risk Engine
- Transformation of Data at the integration layer to be acceptable by Fraud Engine
- Testing of Business SOPs, disaster recovery and Business Continuity Plan.

1.2 Objective

A primary objective of testing application systems is to: ***assure that the system meets the full requirements, including quality.***

- Identify the scope of testable requirements for Card Payment from Merchant
- Establish and maintain a well-founded plan for performing and managing the testing activities
- Arrive at a Estimates and a Delivery Plan for performing test activities within scrum model
- Define test approach based on identified tests
- Establish a structure for all phase of testing

Any changes, additions, or deletions to the requirements document, Functional Specification, or Design Specification will be documented and tested at the highest level of quality allowed within the remaining time of the project and within the ability of the test team.

1.3 Project Overview

We see that the external payment from Merchants and the open banking is trending in the Banking Payment sector.

In order to grow with the new features in the payment world as part of this project we are enabling the customers to make payment from a merchant via Card, process the transaction with the customer details available in the bank data base and send back the response to the payment merchant regarding the transaction details.

1.4 Roles and Responsibilities

Below listed functions are for testing phase.

1.4.1 Tester

Quality Engineer/Test Analyst team are responsible for the below mentioned activities

(a) Contribute to Use case

(b) Contribute to develop and execution of the development test scripts through review

(c) Conduct Full User Acceptance, regression, and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts and reporting test results

1.4.2 Testing Process Management Team

The team responsible to manage the entire testing process, workflow and quality management with activities and responsibilities to:

(a) Monitor and manage testing integrity and Support testing activities

(b) Coordinate activities across teams such as DB team, Fraud team etc

1.5 Assumptions for Test Execution

Below are some minimum assumptions that have been considered.
New assumptions may also be added that are reasoned to be suitable to the project.

- For User Acceptance testing, the team has completed unit, system and integration testing and met all the Requirement's (including quality requirements)

- User Acceptance testing will be conducted by End-users
- Test results will be reported to stakeholders and scrum team on daily basis using JIRA/Confluence. Failed scripts and defect list from JIRA with evidence will be sent to Developer directly.
- Use cases have been developed by testers for User Acceptance testing. Use cases are approved by test lead.
- Test Team will support and provide appropriate guidance to testers and Developers to conduct testing
- Major dependencies should be reported immediately after the testing kickoff meeting.

1.6 Definition of Ready

The below are the DOR for Card Payment from Merchant stories implementation

- Test Cases should be available in JIRA
- Test data mapping should be completed for each test case
- Test Environment should be up and running without any issues
- Back end changes should be deployed into the test environment

1.7 Definition of Done

The below are the DOD for Card Payment from Merchant stories implementation

- Planned test cases should be executed successfully
- No outstanding Priority 1,2,3 or 4 defects and no outstanding Severity 1, 2, 3 or 4 defects
- In case of open defects , approval from business is required to accept the risk and to pass results to next phase of testing and into production

2 Test Methodology

Agile methodology – Scrum Framework

2.1 Purpose

2.1.1 Overview

The purpose of the Test Plan is to achieve the following:

- Define testing strategies for each area and sub-area to include all the functional and quality (non-functional) requirements.
- Divide Design Spec into testable areas and sub-areas. Be sure to also identify and include areas that are to be omitted (not tested) also.

- Define bug-tracking procedures.
- Identify testing risks.
- Identify required resources and related information.
- Provide testing Schedule.

2.1.2 Test Strategy

This test approach defines high-level description of the Card Payment Integration between the merchant and the bank

Module	Testing Type	Features to be tested
Payment UI	UI/Functional testing	A payment page will have following mandatory fields- Customer ID,Transaction ID,Amount and Merchant ID -Payment with all the mandatory field populated gets successfully processed and acknowledgment message is displayed for the transaction -Payment with missing mandatory field cannot be submitted and hard error message is displayed for the transaction
Payment UI/DW to Integration Layer	Integration Testing	After successful submission of payment via payment UI, below actions:- - A payment page will trigger the mandatory field data like customer ID,Transaction ID,Amount and Merchant ID to the integration layer [Data gets stored in MQ and processed] - DW will automatically trigger the enriched data - Customer DOB, Customer Post Code and Merchant Post Code against the payment record received by integration layer
Integration Layer to FDE Fraud Detection Engine	Integration Testing	Integration layer receives the data from both, Payment UI and DW which gets converted into a acceptable format and sent to Fraud Detection Engine for processing via TCP Data is received by FDE which gets validated,processed and the appropriate response is sent back to Integration layer
Payment UI/Integration Layer/DW/FDE	UI/Functional/SIT/ST Testing	**Positive -** Payment get successfully processed in FDE and acknowledgement is displayed in payment UI for below - Customer is above 18 { and } All mandatory fields from both DW and payment UI are recevied {and} Merchant's PO code falls under NSW **Negative -** Payment gets rejected by FDE and declined message is displayed in the Payment UI for the below condition:- Condition 1 - customer is minor [Under 18] and the amount processed is greater than 5000 Condition 2 - Payment submitted for amount greater than 10000 with missing customer PO code [Ideally ,Data should be received by integration layer from DW] Condition 3 - Payment submitted for customer with missing

		DOB [Ideally ,Data should be received by integration layer from DW] Condition 4 - If Merchant PO code does not fall under NSW area and the amount submitted is greater than 20000 Condition 5 - Condition 1/ Condition 2/ Condition 3 [And/OR] Condition 4

Integration Point 1: Passing the details from Merchant Website to the Banks Integration layer via MQ in order to process the Card Payment initiated by the Customer

Integration Point 2: Passing the customer details received from Merchant and Bank's data warehouse required for scoring the customer through TCP to Fraud Decision engine and sending back the payment response through the Integration Layer

Phase	Area of Change	Teams	Test Activities	Remarks
Point 1	**thirdPartyPayment API** should be ready for Card Payment merchant to interact with Bank back end systems	Feature Team	• Ensure the API is working as expected before integration using mock responses • Once proved, integrate the system with the third party merchant and verify if transaction details are sent through the request	
Point 2	**retrieveCustomerDetails API** should be ready to retrieve the customer details from DW	Feature team	• Ensure API is working as expected and we are able to retrieve Customer and Merchant details from DW required to fulfill the transaction	
Point 3	**trasactionScoring API** should be built with valid rules in order to decline or pass the transaction based on the details sent to the fraud risk engine	Fraud and Crime Team	• Send Customer details to **Fraud risk engine** using **transactionScoring** API in a format accepted by Fraud Risk Engine	• **transactionScoring API** should be fully developed and tested by Fraud

			• **transactionScoring API** should send the response back to the integration layer with valid response	team and provided to the feature team to use • **The response sent by transactionScoring API** is then sent to Merchant payment

The document includes the below test items for each of the integration layers involved in the transaction.
Test Items:

- Test Execution Procedures
- Test Deliverables
- Test Data Management
- Test Schedule
- Test Environment

2.1.3 Unit Test

The purpose of unit testing is to allow the **developer** to confirm the functionality provided by a single unit or component of code. Additionally, wherein one component cannot function without interacting with another component, the test shall include limited interactions.
Unit testing shall consist of the following:

- Static testing – Conducting "walkthroughs" and reviews of the design and coded components.
- Basic path testing – Executing path testing based on normal flow.
- Condition/multi-condition testing – Executing path testing based on decision points.
- Data flow testing – Examining the assignment and use of variables in a program.
- Loop testing – Checking the validity of loop constructs.
- Error testing – Executing unexpected error conditions.

2.1.4 Integration Test

Integration testing confirms that each piece of the application interacts as designed and that all functionality is working. Integration testing includes interactions between all layers of an application, including interfaces to other applications, as a complete end-to-end test of the functionality.
Integration testing shall consist of the following:

- Verifying links between internal application components.

- Focusing on complete end-to-end processing of programs, threads, and transactions.
- Boundary value analysis (testing modules by supplying input values within, at, and beyond the specified boundaries).
- Comparison testing (comparing output of system under test with another reference system).
- Ensuring traceability to requirements, use cases, user interface (UI) design, and test objectives.
- Testing each business function end-to-end through the application, including positive and negative tests.

2.1.5 Usability Testing

The purpose of usability testing is to ensure that the new components and features will function in a manner that is acceptable to the customer.
Development will typically create a non-functioning prototype of the UI components to evaluate the proposed design. Testing will review the findings and provide the project team with its evaluation of the impact these changes will have on the testing process and to the project as a whole.

2.1.6 Iteration/Regression Testing

During the repeated cycles of identifying bugs and taking receipt of new builds (containing bug fix code changes), there are several processes which are common to this phase across all projects. These include the various types of tests: functionality, performance, stress, configuration, etc. There is also the process of communicating results from testing and ensuring that new drops/iterations contain stable fixes (regression). The project should plan for a minimum of 2-3 cycles of testing (drops/iterations of new builds).

At each iteration, a debriefing should be held. Specifically, the report must show that to the best degree achievable during the iteration testing phase, all identified severity 1 and severity 2 bugs have been communicated and addressed. At a minimum, all priority 1 and priority 2 bugs should be resolved prior to entering the beta phase.

2.1.7 User Acceptance Test

The purpose of user acceptance testing (UAT) is to simulate the business environment and emphasize security, documentation, and regression tests. UAT may be performed by a third party in cases where the general user community is large and may provide different goals and objectives for acceptance testing requirements.
UAT shall be conducted to gain acceptance of all functionality from the user community. UAT shall verify that the system meets user requirements as specified.

2.1.8 Pre Prod Test

The purpose of operational readiness testing is to identify any potential issues with the production environment setup before users access the system.
Operational readiness testing shall verify that the application move from the acceptance environment to the production environment was successful.

2.1.9 Beta Test

In beta testing, a small number of experienced users try the product in a production mode and report defects and deficiencies. The purpose of beta testing is to identify suggested improvements into a general release for the larger user community.

Beta testing shall consider the following issues:
- Proper identification of the beta testing group.
- Specific areas for which feedback is requested.
- Specific areas for which feedback is not requested.

2.1.10 Testing completeness Criteria

Release for production can occur only after the successful completion of the application under test throughout all of the phases and milestones previously discussed above.

The milestone target is to place the release/app (build) into production after it has been shown that the app has reached a level of stability that meets or exceeds the client expectations as defined in the Requirements, Functional Spec.,.

2.1.1 Non-Functional Testing

- Security testing
  - Penetration testing
  - Static Application Security testing
  - Dynamic Application Security testing
- Performance Testing
- Load Testing
- Stress Testing
- Web tagging/MI

The above mentioned Non Functional tests should be completed before production release

2.2 Bug Regression

Bug Regression will be a central tenant throughout all testing phases.
All bugs that are resolved as “Fixed, Needs Re-Testing” will be regressed when Testing team is notified of the new drop containing the fixes. When a bug passes regression it will be considered “Closed, Fixed”. **When a Severity 1 bug fails regression, Testing team should also put out an immediate email to development.** The Test Lead will be responsible for tracking and reporting to development and product management the status of regression testing.

2.3 Bug Triage

The Test Lead, Product Manager, and Development Lead should all be involved in these triage meetings. The purpose of the triage is to determine the type of resolution for each bug and to

prioritize and determine a schedule for all "To Be Fixed Bugs'. Development will then assign the bugs to the appropriate person for fixing and report the resolution of each bug back into the JIRA bug tracker system.

2.4 Test Completeness

Testing will be considered complete when the following conditions have been met:

2.4.1 Standard Conditions:

- When Testers and Developers, agree that testing is complete, the app is stable, and agree that the application meets functional requirements.
- Script execution of all test cases in all areas have passed.
- Automated test cases have in all areas have passed.
- All priority 1 and 2 bugs have been resolved and closed.
- Each test area has been signed off as completed by the Test Lead.
- 50% of all resolved severity 1 and 2 bugs have been successfully re-regressed as final validation.
- Ad hoc testing in all areas has been completed.

2.4.2 Bug Reporting & Triage Conditions:

- Bug find rate indicates a decreasing trend prior to Zero Bug Rate (no new Sev. 1/2/3 bugs found).
- Bug find rate remains at 0 new bugs found (Severity 1/2/3) despite a constant test effort across 3 or more days.
- Bug severity distribution has changed to a steady decrease in Sev. 1 and 2 bugs discovered.
- No 'Must Fix' bugs remaining prior despite sustained testing.

3 Test Deliverables

Test deliverables fall into three basic categories: Documents, Test Cases / Bug Write-ups, and Reports

3.1 Deliverables Matrix

Below is the list of artifacts that are process driven and should be produced during the testing lifecycle.

Deliverable
Documents
Test Approach
→ Test Plan
→ Test Schedule

→ Test Specifications
Test Case / Bug Write-Ups
Test Cases / Results
Test Coverage Reports
JIRA Bug tracker for bug reporting
Reports
Test results report
Test Final Report - Sign-Off

3.2 Defect Tracking & Debugging

3.2.1 Defect reporting using JIRA

ALL defects should be logged using 'JIRA', to address and debug defects. Testers are also requested to send a daily defect report to the developer

Debugging should be based on Priority – High > Medium > Low, these priorities are set by the Testers and are based on how critical is the test script in terms of dependency and mainly based on use case scenario.

3.3 Reports

The Test Lead will be responsible for writing and disseminating the following reports to appropriate project personnel as required.

3.3.1 Testing status reports

A daily, weekly or bi-weekly status report will be provided by the Test Lead to project personnel. This report will summarize weekly testing activities, issues, risks, bug counts, test case coverage, and other relevant metrics.

3.3.2 Phase Completion Reports

When each phase of testing is completed, the Test Lead will distribute a Phase Completion Report to the Product manager, Development Lead, and Program Manager for review and sign-off.

3.3.3 Test Final Report - Sign-Off

A Final Test Report will be issued by the Test Lead. It will certify as to the extent to which testing has actually completed (test case coverage report suggested)

4 Resource & Environment Needs

4.1 Testing Tools

- JIRA
- Confluence
- Post Man
- SOAP UI
- Splunk
- Zephyr test case
- Webdriver IO framework in place for Automation

4.2 Test Environment

- Lower environment
- System Integration Environment
- UAT Environment
- Pre Prod Environment
- Prod Environment
- Third Party Environment to access DW, Fraud decision making etc.

4.3 Bug Severity and Priority Definition

The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs.

4.3.1 Severity List

The tester entering a bug into JIRA is also responsible for entering the bug Severity.

Severity ID	Severity	Severity Description
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

4.3.2 Priority List

Priority	Priority Level	Priority Description
5	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.

4	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
2	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
1	Trivial	Enhancements/ Good to have features incorporated- just are out of the current scope.

4.4 Bug Reporting

JIRA is used as a Bug Tracking tool and the same would be mapped with relevant user stories which are impacted.

5 Dependencies, Risks and Assumptions

5.1 Dependencies

Dependency	Potential Impact of Dependency	Owners
Fraud Risk Engine rules should be activated to score the customer based on the input received	Unable to proceed with end to end testing without a valid response from the Fraud Risk Engine	Security/Fraud Team
Increase in TPS for the Data retrieval API should be agreed by the DB team	Response time would be impacted	Retrieval API Owner

5.2 Risk

Risk	Potential Impact of Risk	Owners
Environment stability for third party APIs	Unable to proceed with end to end testing without a stake environment	Third Party team

5.3 Assumptions

Assumptions	Potential Impact of Assumption	Owners

Assumptions	Potential Impact of Assumption	Owners
Test Data/Test Environment is available to kick off the testing	Without Test Data/Test Environment testing could not be started	Feature Team

6 Approvals Section

Role	Approver Name	Status
Product Owner	PO	--
Scrum Master	SM	--
Business Analyst	BA	--
Test Lead	TL	--

7 Terms/Acronyms

The below terms are used as examples, please add/remove any terms relevant to the document.

TERM/ACRONYM	DEFINITION
API	Application Program Interface
BCP	Business Continuity Plan
CAT	Client Acceptance Testing
End-to End Testing	Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the same set of data from beginning to end, as intended.
N/A	Not Applicable
QA	Quality Assurance
RTM	Requirements Traceability Matrix
SME	Subject Matter Expert
SOP	Standard Operating Procedure
TBD	To Be Determined
TSR	Test Summary Report
DW	Data Warehouse
TPS	Transaction Per Second