Freightsy-Equitime

Test Plan Document

Version: Draft

Abstract

This document provides an overview of the project, details of test planning activities and a list of testing deliverables over a six months' time frame

TEHREEM FATIMA

1-31-2022

Version History

Version	By	Date	Outline
1.0	Tehreem Fatima	01-31-2022	Test Plan - Draft

Contents

1. Introduction

1.1. Purpose

This test plan describes the testing approach and overall framework that will drive the testing of the Equitime application. The document introduces:

- **Test Strategy**: Rules on which testing will be based on, including the givens of the project (e.g.: start / end dates, objectives, assumptions); description of the process to set up a valid test (e.g.: entry / exit criteria, creation of test cases, specific tasks to perform, scheduling, data strategy).
- **Execution Strategy:** Describes how the test will be performed and process to identify and report defects and implement fixes.
- **Bugs Reporting:** Process of reporting and tracking bugs and regression after bug fixes
- **Test Management:** Process to handle the logistics of the test and all the events that come up during execution (e.g.: communications, escalation procedures, team members)

1.2. Project Background

Equitime is a critical internal software application that supports the end-to-end management of customer contracts, including finance. Application contains the database of the Companies that joins Equitime to create Contracts with other companies. When the authenticated User is added to the application then he can see the company's contracts.

2. Test Strategy

2.1. Test Objectives

The objective of the tests is to verify that "the implementation and integration of the "Equitime Application" works according to the specifications.

The tests will do the following,

- Finding defects, which are induced during the development?
- Prevention of bugs through regression testing.
- To make sure that the result meets the business and user requirements.
- To gain the confidence of the users by providing them a quality product.

2.2. Test Assumptions

Key Assumptions

- Testers should be provided a test environment with different user roles.
- For QA, user, company and the contract details should be given prior to start of functional testing

General

- Exploratory Testing would be carried out once the build is ready for testing.

- Performance testing should be performed to check the load on the application when users are logged in, when new users are added or when new data is added.
- All the defects would come along with a snapshot JPEG format and recorded videos.
- The Test Team will be provided with access to Test environment.
- The Test Team assumes all necessary inputs required during Test design and execution will be supported by Project Manager/ Product Owner appropriately.
- Test case design activities will be performed by QA personnel and once the QA plan is ready it should be peer reviewed.
- Test environment and preparation activities will be owned by Development Team.
- Development team will provide Defect fix plans based on the Defect meetings during each cycle to plan. The same will be informed to Test team prior to start of Defect fix cycles.
- Project Manager/ Product Owner will review all Test cases prepared by Test Team prior to start of Test execution.
- The defects will be tracked through assigned tool. Any defect fixes planned will be shared with Test Team prior to applying the fixes on the Test environment.
- There must be a feature freeze date (at-least 3 days) before the release date and only bugs will be fixed in those days till the release. This will give testers an ample time to execute their test cases along with regression plan after bug fixes
- Developers should test their push with QA before merge

Functional Testing

- In functional testing, testing team will be given access to all roles (If there are multiple roles for admin and users.
- The testing team will be provided with the companies, users and the contract details.
- The testing team will be performing Functional testing on “Equitime Application”.

2.3. Test Principles

- Testing will be focused on meeting the business objectives, cost efficiency, and quality.
- Testing processes will be well defined, yet flexible, with the ability to change as needed.
- Testing activities will build upon previous stages to avoid redundancy or duplication of effort.
- Testing will be a repeatable, quantifiable, and measurable activity.
- Testing will be divided into distinct phases (Sprint planning and during sprint phase), each with clearly defined objectives and goals.
- There will be entrance and exit criteria.

2.4. Scope and Levels of Testing

2.4.1. Exploratory Testing

The **purpose** of this kind of tests is to make sure **critical defects** are removed before the next levels of testing can start. The testing will be performed by the **testers** at the **beginning** of each cycle **without** any test scripts and documentation.

Issues found during this phase would be reported immediately and should be fixed for regression testing by the testers.

2.4.2. Functional Testing

The **purpose** of functional testing will be performed to check the functions of application. The functional testing is carried out by feeding the input and validates the output from the application. The testing will be performed according to functional test cases stored in the **Test Management tool.** The functional tests will commence right **after** the exploratory tests are completed.

2.4.3. Automation Testing

Automation scripts will be created for UI screens and interfaces that are newly created or modified during the development cycle for regression point of view. In spare time, automation script will be created, gradually building up full set of tests.

Where applicable, automated test suites for web services/APIs will be created as well.

2.4.4. Performance Testing

Scripts for Performance Testing will be created to check the load on the application with certain number of users (150 maximum concurrent users as explained in the requirements).

2.5. Testing of New Feature, User Stories

Testing of the new features can be divided into two phases:

2.5.1. Sprint Planning Phase:

- Testers must be included in the sprint planning meetings and Product Owner creates task for new features and user stories with detailed and adequate description
- PO along with QA should be responsible for writing the acceptance criteria
- QA writes the Manual Test Cases based upon the acceptance criteria

2.5.2. Two Weeks Sprint Phase:

- Steps of the Test Cases should be well written before the ticket come for the QA
- Smoke and Sanity Test Plans should be updated according to the tickets in sprint
- When Developer moves the ticket to QA (Review) then Tester, verifies the ticket according to the acceptance criteria and the steps written in test cases.
- Test Executions should be maintained according to the test plan and should cover all the tickets inside the sprint
- Blocker bugs should be directly assigned to PO and must be fixed immediately
- If there is a fix from the bug, tester execute the Regression Test plan and give a go ahead for the release build
- After every sprint, the left-over tests should be reviewed by the PO and Dev Lead and then the linked tickets should be worked upon
- Once the build is stable and manual test cases are well documented, subsequent test cases should be automated.

2.6. Features to be Tested

All the features of Equitime application which were defined in software requirement specs needs to tested. An example of some scenarios is explained below

ID	Scenarios	Users/Entities	Description
01	Only Valid Users should be logged in	Company Employee (User)	• A user belongs to the company that has valid credential details with correct email address and password • User having invalid credentials should be authenticated and Error message occurs on the login page
02	Details of Valid contracts should be available to authenticated users	Company Employee (User)	• When the user is authenticated, then he should be directed to the Company Dashboard • All the details including all related company contracts, should be available on the dashboard page
03	Admin should be able to add new users to the project	Admin	• When the new company wants to onboard to the company then Admin of the application should be able to add the users • Here we will test the Admin and the User Roles
04	Admin should be able to create new contracts with the companies upon users' requests	Admin	• When one company wants to create contracts with another company then admin should be able to create the contract documents for them • Here we will also test the Admin and the User Roles
05	Contracts should be stored correctly in the Database	Admin	• Contract documents which are added by the companies should be stored in the database with unique ids • Here we perform the database testing

3. Execution Strategy

3.1. Entry and Exit criteria

Entry and exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation. All this is input to the project manager for a final "go-no go" decision.

3.1.1. Entry criteria

The entry criteria refer to the desirable conditions in order to start test execution; only the migration of the code and fixes need to be assessed at the end of each cycle.

- **Entry criteria to start the execution phase of the test:** the activities listed in the Test Planning section of the schedule are 100% completed.
- **Entry criteria to start each cycle:** the activities listed in the Test Execution section of the schedule are 100% completed at each cycle.

3.1.2. Exit criteria

The exit criteria are the desirable conditions that need to be met in order proceed with the implementation. The exit criteria are only met when the following conditions are satisfied.

- 100% Test Scripts/cases are executed.
- 95% pass rate of Test Scripts/cases.
- No open Critical and High severity defects.
- 95% of medium severity defects have been closed.
- All remaining defects are either cancelled or documented as Change Requests for a future release.
- All expected and actual results are captured and documented with the test cases.
- All defects logged in JIRA.

3.2. Test Cycles

- There will be two cycles for functional testing. Each cycle will execute all the scripts.
- The objective of the first cycle is to identify any blocking, critical defects, and most of the high defects (**SMOKE Test Execution**)
- The objective of the second cycle is to identify remaining high and medium defects and obtain the results with positive and negative testing scenarios (**SANITY Test Execution**).
- Suites for Smoke, Sanity, and Regression tests will be created in **Test Management tool**. Smoke and Sanity test execution will be performed in cycles for every build provided to the QA team. Regression suite will be executed either on demand or based on the requirements given by the PO to verify the bug fixes and new functionality.

3.2.1. Smoke Testing

Smoke testing is a preliminary level of testing that ensures all basic components of an application are functioning properly and only tests major functions. It gives us the sense if it's sensible to move ahead with exhaustive testing.

For **Equitime Application** smoke tests will cover (But not limited) essentials such as:

- Login of Authenticated Users only.
- Verifying the correct layout and accuracy of all visual elements.
- Contracts creations with the companies.
- Company's contracts should be accessible to the designated users
- Contract Details should be correctly specified
- Logout from the application

The main reason to run smoke tests would be to a get rapid feedback—in only a few minutes or an hour—that the build is essentially sound. The primary purpose would be to get the feedback immediately so that we (testers) don't potentially waste hours to get the same critical answers.

Goals of smoke testing

1. Detect show-stopping bugs much earlier
2. Improve the effectiveness of the QA team
3. Faster troubleshooting of new and regression bugs

Deliverables to Freightsy

To be efficient and purpose-built, a smoke test suite would contain an easily manageable number of tests that execute very quickly—even though it would be fully automated. A good range for the number of tests that we are thinking right now, would be from about 15 to 20 tests. It must be kept in mind that too little or too much coverage might defeat the purpose of this preliminary test suite.

Ideally, each smoke test would meet the following criteria:

- Tests core features only
- The test should be reproducible indefinitely
- Very fast execution
- Should generate few or no false positives

The main aim of **QA Team** would be to automate all the smoke tests and would aim to complete the running of this set of tests in 5 minutes or less.

3.2.2. Sanity Testing

"Sanity testing is a quick and basic test (or set of tests) to determine if a particular application or component is behaving correctly."

At the outset, we need to understand there is a shared territory between the terms "sanity test," "smoke test" and "regression test." In many ways, you could see "sanity test" and "smoke test" as synonyms [But it is not always the case]. Sanity tests are often fewer in number and more focused than regression tests.

Goals of Sanity Testing

Some goals of sanity tests would be:

1- **Simple**, easily designed and performed.
2- **Pre-defined checklists or exploratory** testing based on intuition and experience.
3- **Comprehensive enough** to give a basic assessment of behavior, which checks the positive and negative scenarios
4- **Performed** quickly for the purpose of quick feedback.

3.2.3. Regression Testing

"Regression testing involves repetitive tests that aims to verify that previously working software still works after changes to other parts. Regression testing shall ensure that nothing has been affected or destroyed"

*For **Equitime Application** Regression testing will be done (but not limited) when:*

- New functionalities are added or when extending the features of the software
- In case of change in requirements
- When there is a bug fix
- When there are performance issues
- In case of environment changes

Goals of Regression Testing

1. To increase the chances of detecting bugs early
2. To help in finding an unwanted side effect that might have been introduced due to a recent change
3. Ensure better performing software
4. To verify that the code changes have not re-introduce old bugs

Selection criteria of Test cases for Regression suites

Based on the following points **QA Team** would create regression suite.

- Test cases which have frequent defects
- Complex test cases
- Integration test cases
- Test cases which cover the core functionality of a product
- Functionalities which are frequently used
- Test cases which frequently fail
- Test cases of all bugs

There would be at-least three suites (as mentioned above) in the test management tool, each of them executed depending upon the situation and time constraints.

3.2.4. Performance Testing

*"*Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload.*"*

*For **Equitime Application** Performance testing will be done (but not limited) when:*

- New Users or companies are added in the system
- When new products are added in the system
- When the reports are exported to PDF format

Goals of Performance Testing

1. Discovering bottlenecks before deployment
2. Enhance the scalability of a system
3. Ensure better performing software

3.3. Validation and Defect Management

- It is expected that the testers will execute all the test cases/ test scripts but the testing process will not be limited to those scripts only. Testers can perform further tests if and when required.
- All the bugs will be logged and tracked through Project Management Tool.
- It is responsibility of the tester to log the bugs in Project Management Tool.

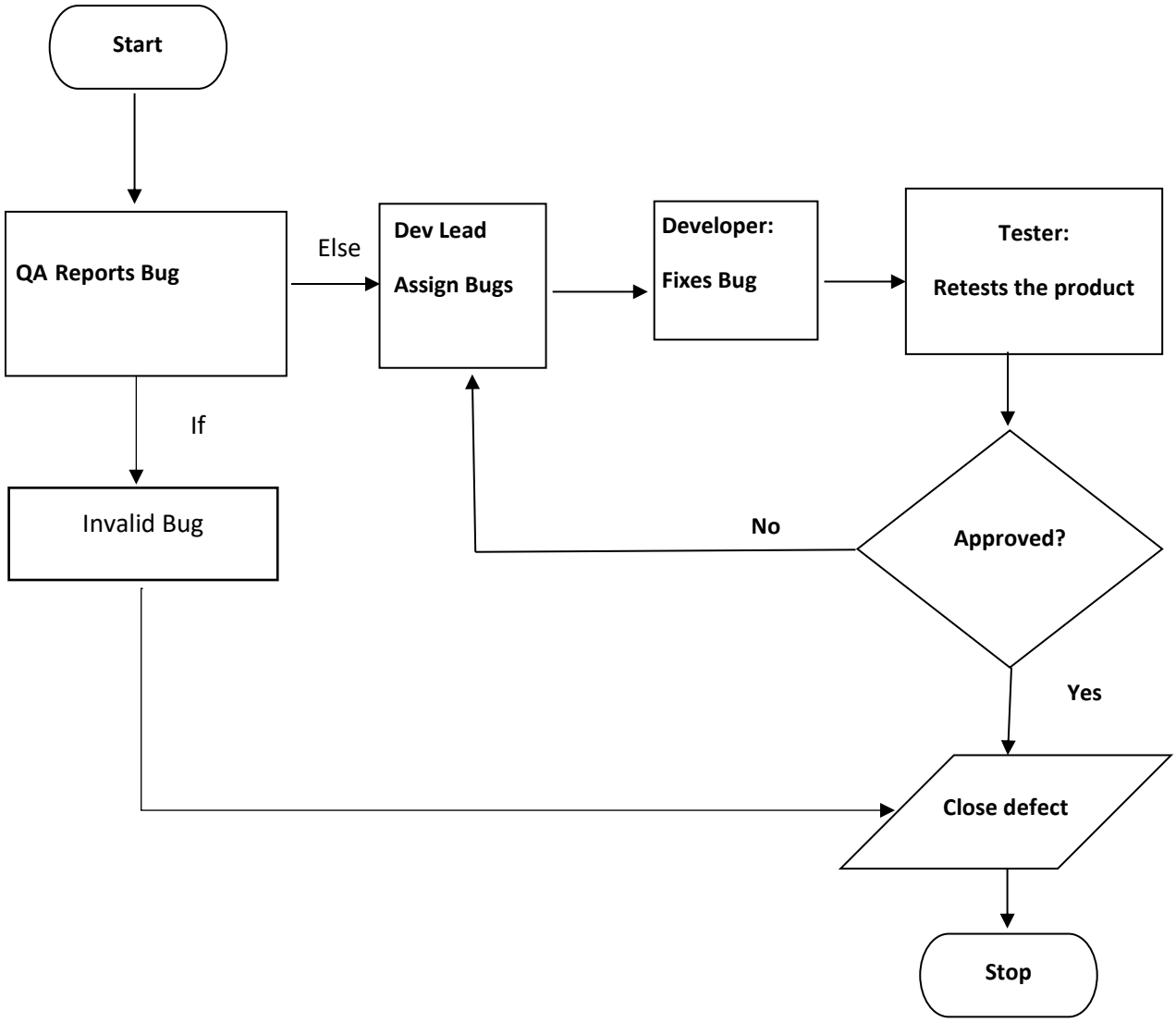
4. Bugs Tracking and Reporting

4.1. Bug Reporting

Tester reports the bug and add as much information as possible for better understanding by the developer. All bugs must follow the same template, which includes the following items:

- Title of the bug
- Environment
    - Operating System
    - Device
    - Account used (Admin or User)
- Steps to reproduce
- Expected Result
- Actual Result
- Visual Proof (screenshots, videos, text)
- Severity (to be decided by the Tester)
    - Blocker
    - Critical
    - High
    - Medium
    - Low
- Priority (to be decided by the PO or Dev Lead)

4.2. Bug Tracking

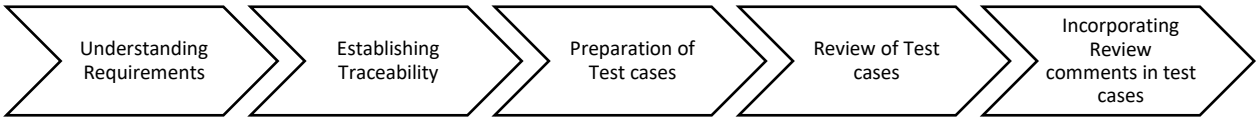


5. Test Management Process

5.1. Test Management Tool

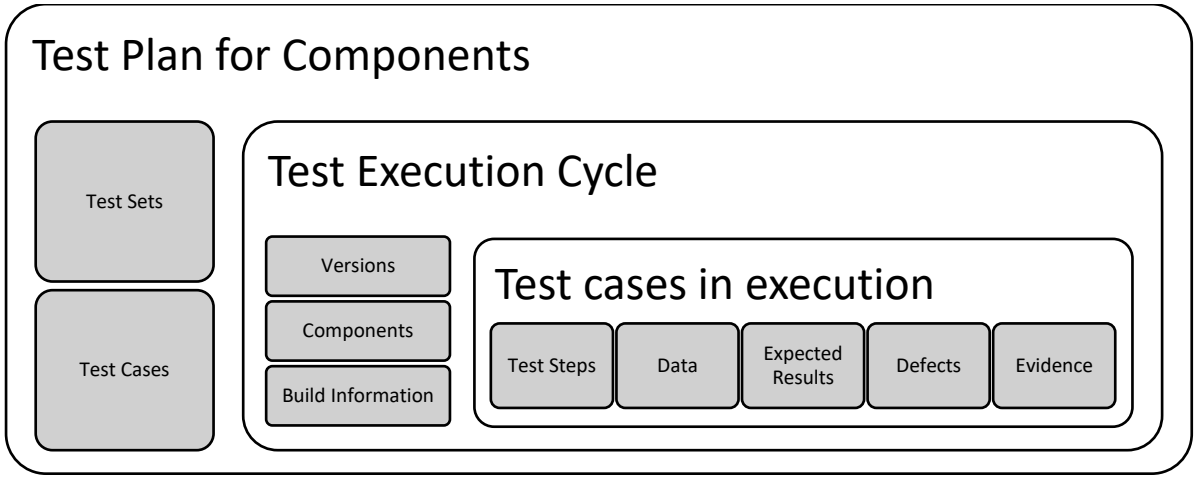
Test Management tool would be used for logging test cases, preparing test plans and cycles while the same tool would be used for reporting. In case of more than one tester each tester will be assigned the test cases which they would execute.

5.2. Test Design Process



5.3. Test Execution Process

5.4. Test Execution Management



6. Risk and Issue

Following are the risk/issues and mitigations:

Risk/Issue	Mitigation
Delay in the form of unforeseen hindrances can take longer than expected.	Set the priority of the story and its sub-tasks, top priority should take out first. PO must set the priority of task during sprint planning
Web service/ API automation	If required, should clearly mention in the story along with request sample and responses.
Delay in UI Automation	Automation will be carried out in spare time of manual testing. Can boost up the speed with extra test resource(s).
The project schedule is too tight; it's hard to complete this project on time	Set Test Priority for each of the test activity

7. Test Environment

Test environment should be setup in a way that it should have its own database, web server, build and deployment mechanism.

8. Resource Planning

8.1. Human Resources

Total resources involved in the project will be:

1. QA Manager -1
2. Manual QA Engineer -2
3. Automation QA Engineer -1
4. Developers (Each developer should write their units tests- Number of developers according to the modules)

Following is the human resource planning of the project:

Sr.	Resources	Tasks
1	Developer	Unit tests
2	QA Manager	Test plan document Manage the whole project Define project directions Acquire appropriate resources
3	Tester/QA	Identifying and describing appropriate test techniques Verify and assess the Test Approach Creates the tests Execute the tests, Log results, Report the defects Reporting & Evaluation
4	Automation QA	Automation script creation White Box Testing Automation testing for UI, API Performance Testing
5	Tester/QA	Test closure activities

8.2. System Resources

Following is the System resource planning of the project:

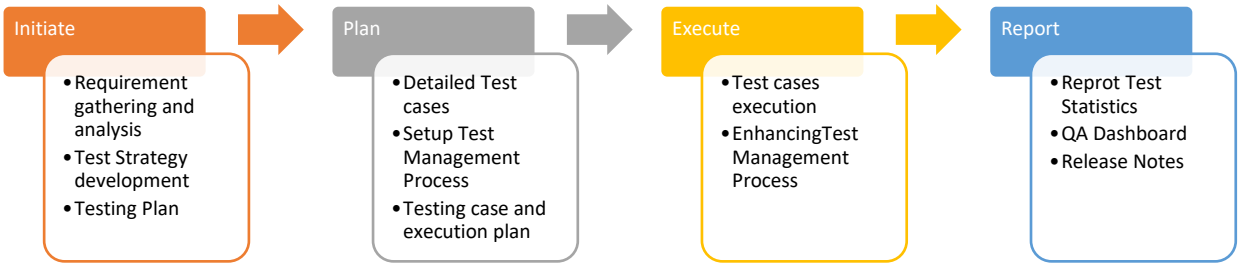
Sr.	Resources	Description
1	Database Server	The documents generated by the application are stored in the database and need to be migrated to a Document Management System where more search/classification functionalities are offered in MySQL database
2	Test Tool	Test Management tool for Manual Testing. Develop a Test tool which can auto generate the test result to the pre-defined form and automated test execution

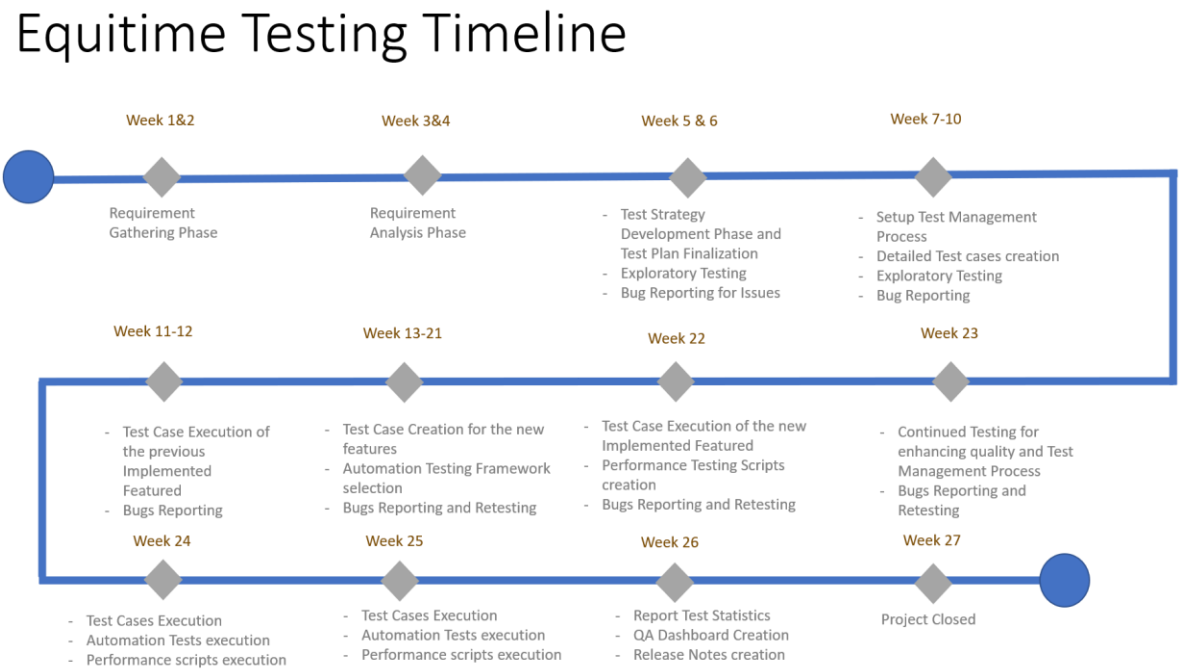
9. Schedule, Estimation and Timeline

Dependent upon the sprint duration, following are the task and their estimation:

9.1. All project task and estimation

Following steps will be involved throughout the project with their estimations on weekly basis





10. Test Deliverables

In the start of project until the Project and Test Management tool is not decided, Test cases will be created in excel sheet in a format that it can be uploaded to test management Tool. All the test case statuses, results, cycles, plans, bugs, bug tracking & monitoring, and related stories/ tasks can be found in tool. However, where necessary, the excel sheet can be shared as test deliverable.

At the end Release Notes, QA Dashboard and the Test Stats along with the testable production software will be delivered.