ZINFI SQA Team Process & Methods

int.prc.002.04 | 04.29.24

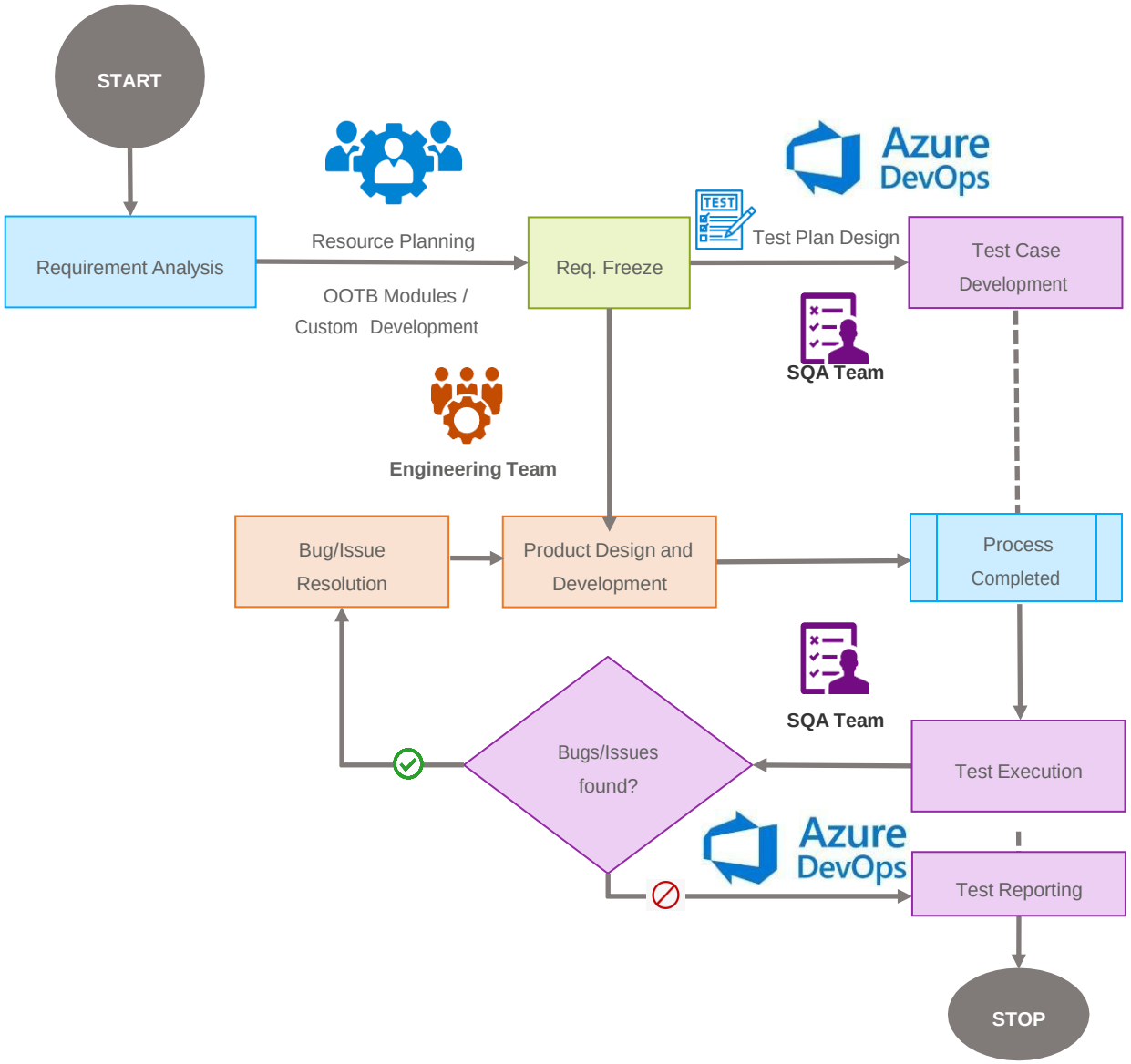
UPM 24.x

ZINFI Confidential & Proprietary
Shared Under NDA

Contents

SQA Process and Delivery Overview

We follow the following standard QA process within our organization. The purpose of this plan is to illustrate the SQA tasks and responsibilities; and guidelines followed to perform the SQA activities; and summarizes the tools, techniques, and methodologies to support SQA activities, and SQA reporting. Process-oriented and focused on preventing software defects, our software quality assurance (QA) process, reaches beyond mere bug detection. The standard flow as well as the repository/tools used are depicted below.



 |

Process Summary

Requirements Analysis

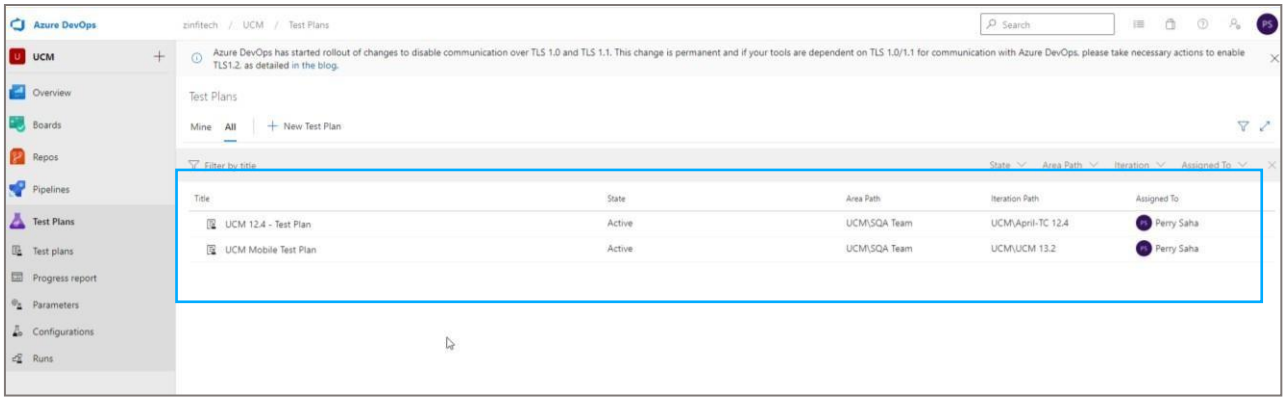
ZINFI uses the Project Management Tool “Teamwork” as our SRS repository. Customizations suggested by Clients are summarized by Project Management Team and are uploaded in Teamwork. ZINFI SQA team goes through the Requirement Specifications and designs the Test Plan. The SQA Team is involved in the analysis and clarification of functional and non-functional software requirements and make sure the requirements are clear, consistent, complete, traceable, and testable. Thus, they prevent possible software defects and facilitate upcoming test design activities.

Resources are planned and setup as per the System Requirements specified. OOTB Modules and Modules citing custom development are listed and maintained through Azure DevOps. *(see below for more details on this.)*

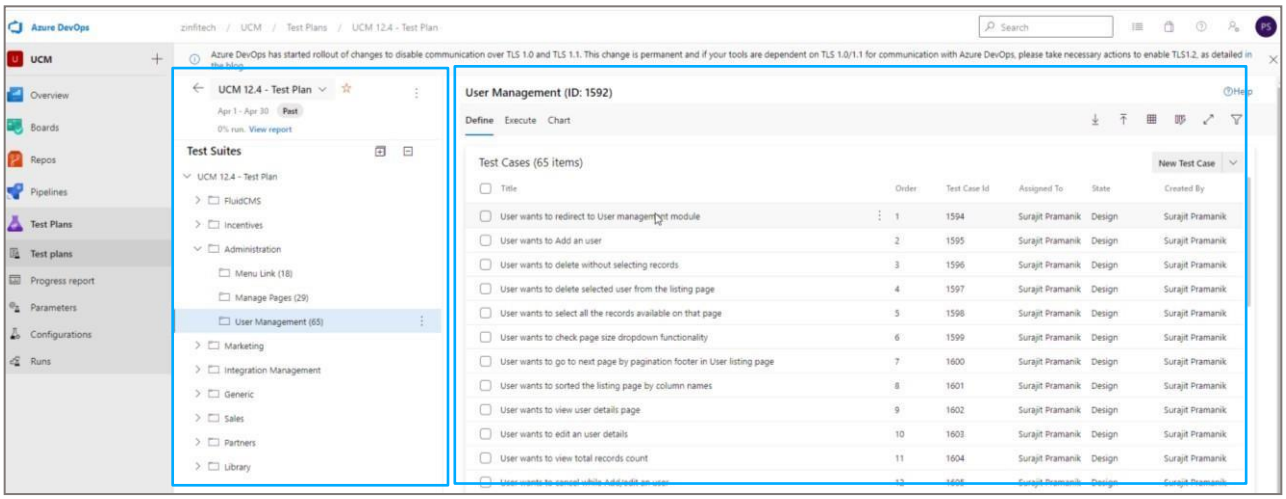
Test Plan Design

ZINFI SQA Team uses the knowledge gained at the requirements analysis stage as a basis for test planning. The Test Plan contains a test strategy and cover a testing scope, and deadlines, the types and levels of testing an application requires, bug tracking and reporting procedures, resources and their responsibilities, and other factors.

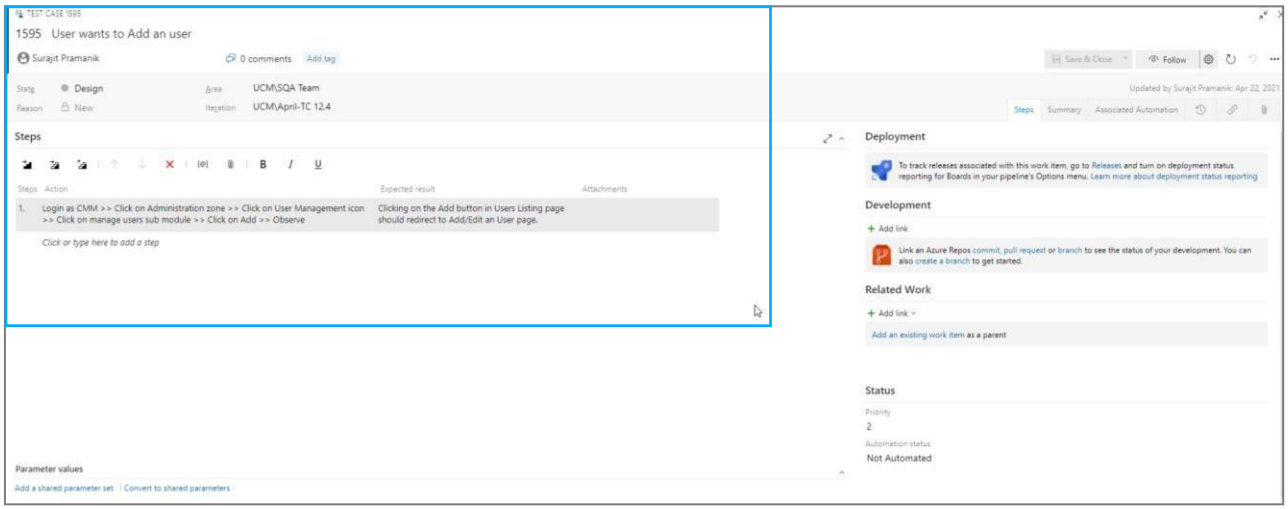
The Test Plan details the objectives, resources, and processes for a specific test process. The plan typically contains a detailed understanding of the process workflow. **A Test Plan is composed of multiple Test Cases.**



A test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve for a specific testing process objective, such as to exercise a particular UPM Module path or to verify compliance with a specific requirement. Test cases underlie testing that is methodical rather than haphazard. A collection of test cases can be built to produce the desired coverage of the Test Plan being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent regression testing.



A test case is usually a single step, or occasionally a sequence of steps, to test the correct behavior/functionality, features of an application. An expected result or expected outcome is usually given.



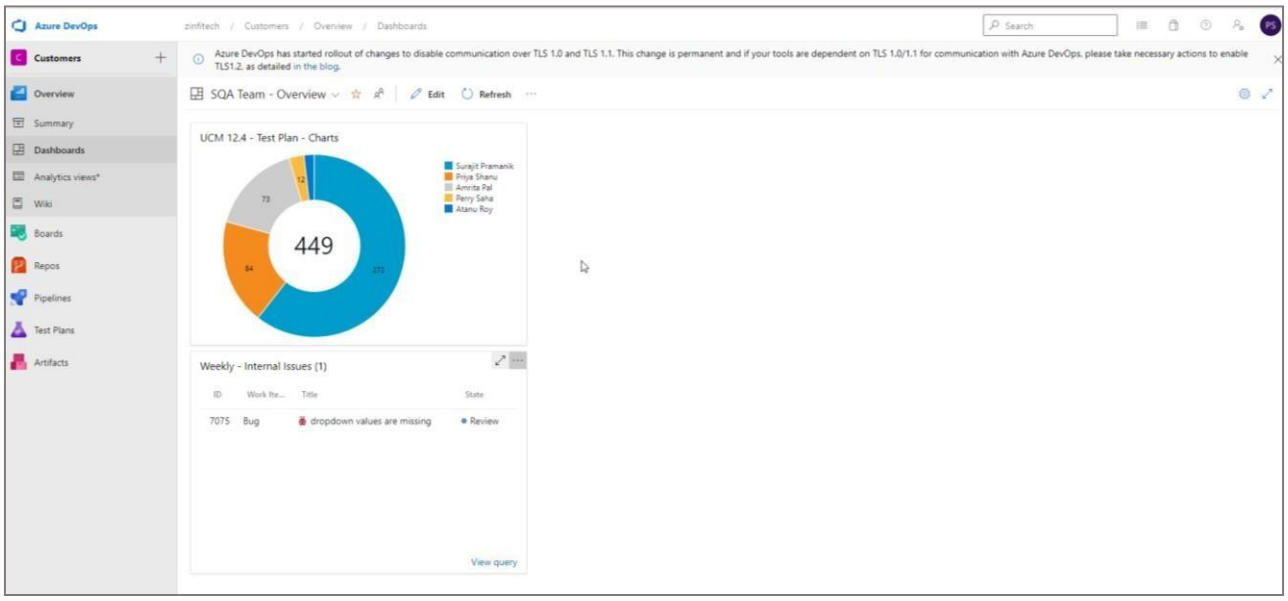
Additional information that may be included:

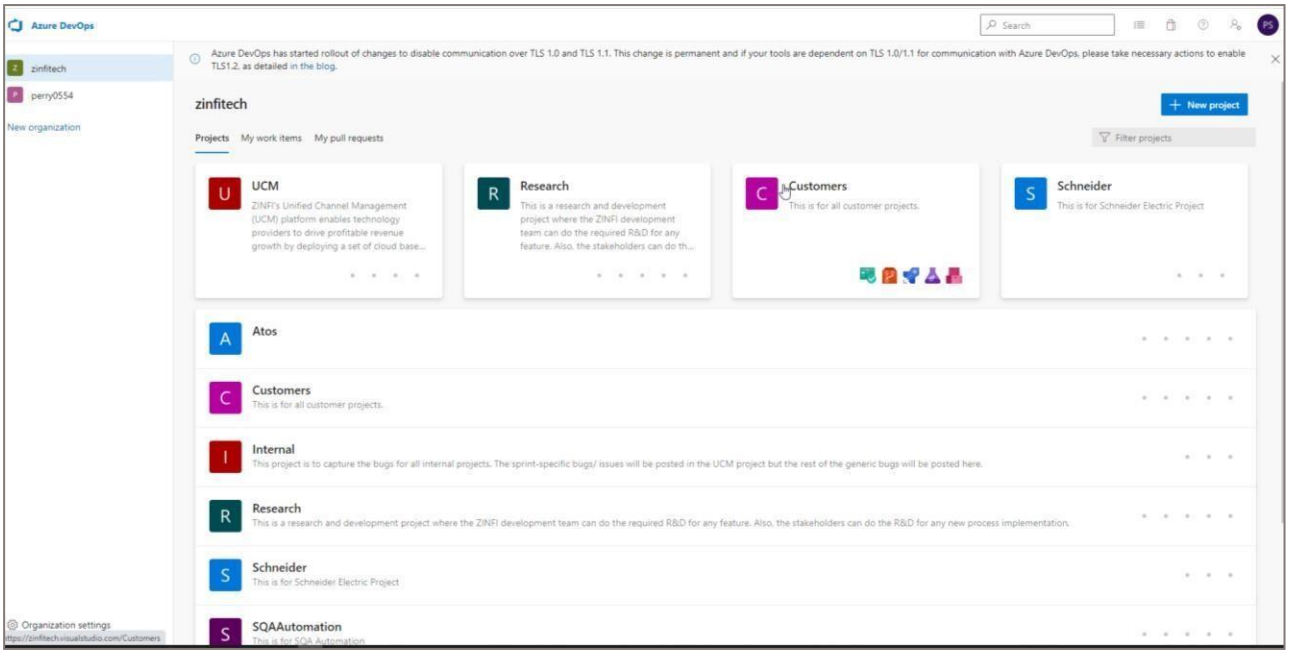
- Test Case ID - This field uniquely identifies a test case.
- Test case Description/Summary - This field describes the test case objective.
- Test steps - In this field, the exact steps are mentioned for performing the test case.
- Pre-requisites - This field specifies the conditions or steps that must be followed before the test steps executions.
- Test category
- Author- Name of the Tester.
- Automation - Whether this test case is automated or not.
- pass/fail and Remarks

 |

Azure DevOps is utilized to create Test Plans and Test Cases.

Azure DevOps Server is a Microsoft product that provides version control (Git), reporting, requirements management, project management (for both agile software development and waterfall teams), automated builds, testing and release management capabilities. It covers the entire application lifecycle and enables DevOps capabilities. Azure DevOps is also utilized as a back-end to integrated development environments (IDEs) for Microsoft Visual Studio.





 |

Azure Test Plans provides rich and powerful tools everyone in the team can use to drive quality and collaboration throughout the development process. The easy-to-use, browser-based test management solution provides all the capabilities required for planned manual testing, user acceptance testing, exploratory testing, and gathering feedback from stakeholders.

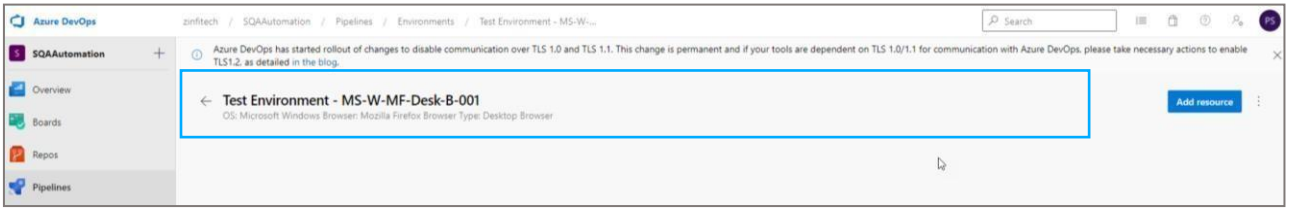
We create and manage test plans and test suites from the Test plans hub. Add one or more test suites—static, requirement-based, or query-based—to the test plans. We define test cases by defining the test steps and optionally the test data to reference. The Grid view for defining test cases supports copy, paste, insert, and delete operations. We assign single or multiple testers to execute tests. View test results and references to a test case across test suites/plans. Within each test case, we specify a set of test steps with their expected outcomes.

Azure Test Plans uses test-specific work item types to plan and author tests. In addition, it provides two test tools to support running tests. The Test plans, Parameters, and Configurations - hubs provide the tools to efficiently create and manage test items, their settings, and configurations. The work item types—Test Plans, Test Suites, Test Cases, Shared Steps, and Shared Parameters—support several explicit links to support requirements tracking and sharing test steps and data across many test cases. Test cases can be assigned as manual or automated.

Test Environments

ZINFI utilizes SIT and UAT environments to ensure that the feature under test, undergoes multiple iterations and utmost sanity testing so that when it is pushed into Production, it appears as almost bug free as possible - we follow the same process for every release and for custom client change requests too.

With Azure test Environments, we define, review, and manage test configurations and variables referenced by test plans. Test Environment configurations provide support for testing our applications on different operating systems, web browsers, and versions. As with shared parameters, test configurations can be shared across multiple test plans.



Test Management and Bug Tracking Tool

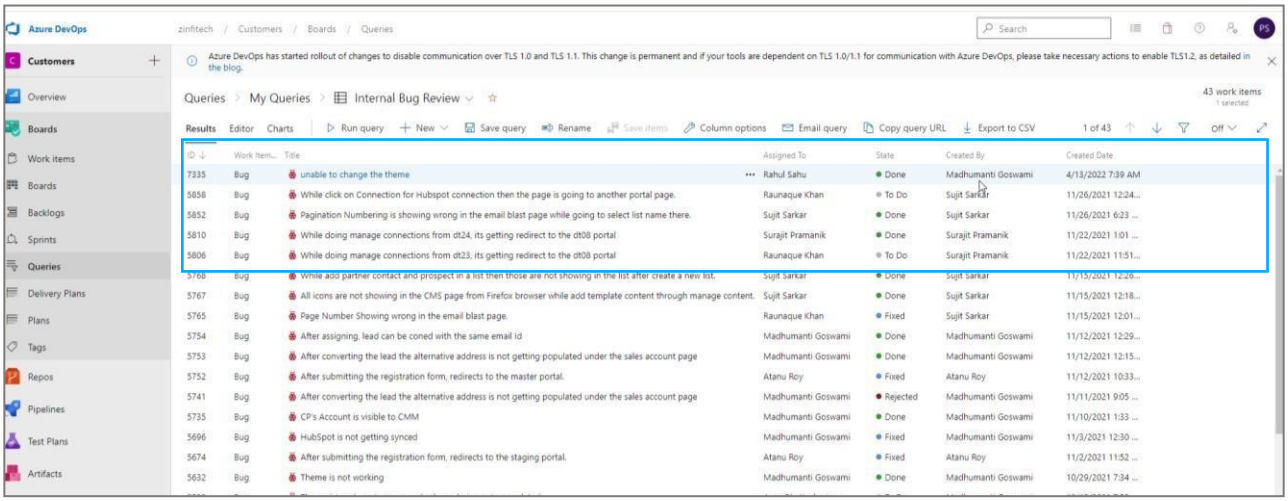
At a minimum, we need a way to capture our software issues, prioritize them, assign them to a team member, and track progress. And we want to manage code defects in ways that align with our Agile practices. Azure DevOps is utilized to design and track every phase of the Test Life Cycle. We are utilizing Azure DevOps platform as the bug tracking tool.

To support these scenarios, Azure Boards provides a specific work item type to track code defects named Bug. Bug work items share all the standard features of other work item types with a few more. The Bug work

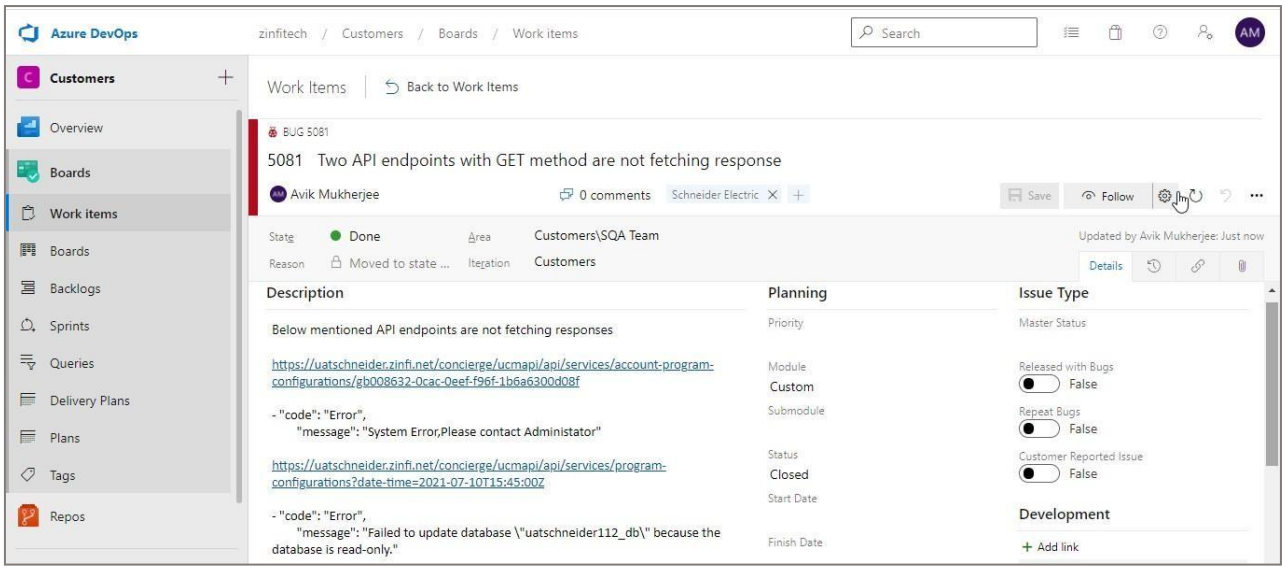
item type uses some bug-specific fields. To capture both the initial issue and ongoing discoveries, we use the following fields:

- Steps to Reproduce - Used to capture enough information so that other team members can fully understand the code defect. Include actions taken to find or reproduce the bug and expected behavior.
- Acceptance Criteria - Provides the criteria to meet before the bug can be closed. Before work begins, we describe the customer acceptance criteria as clearly as possible.

Bugs List with ID, Assignment Status, State and Creation Details:

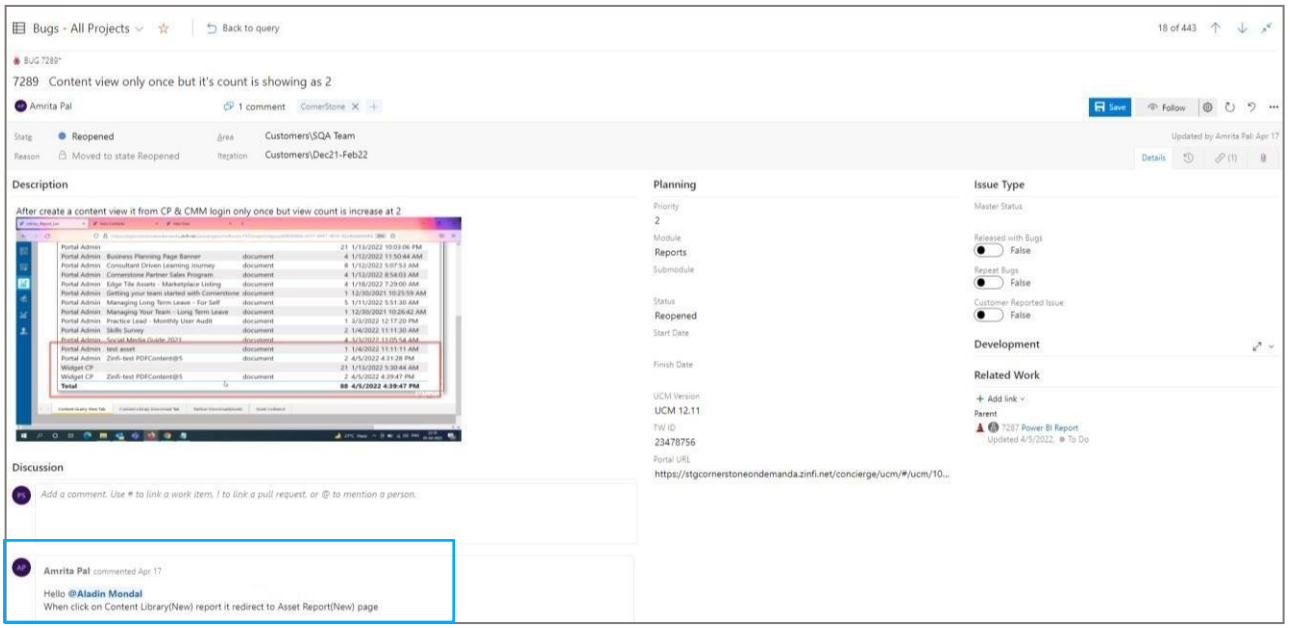


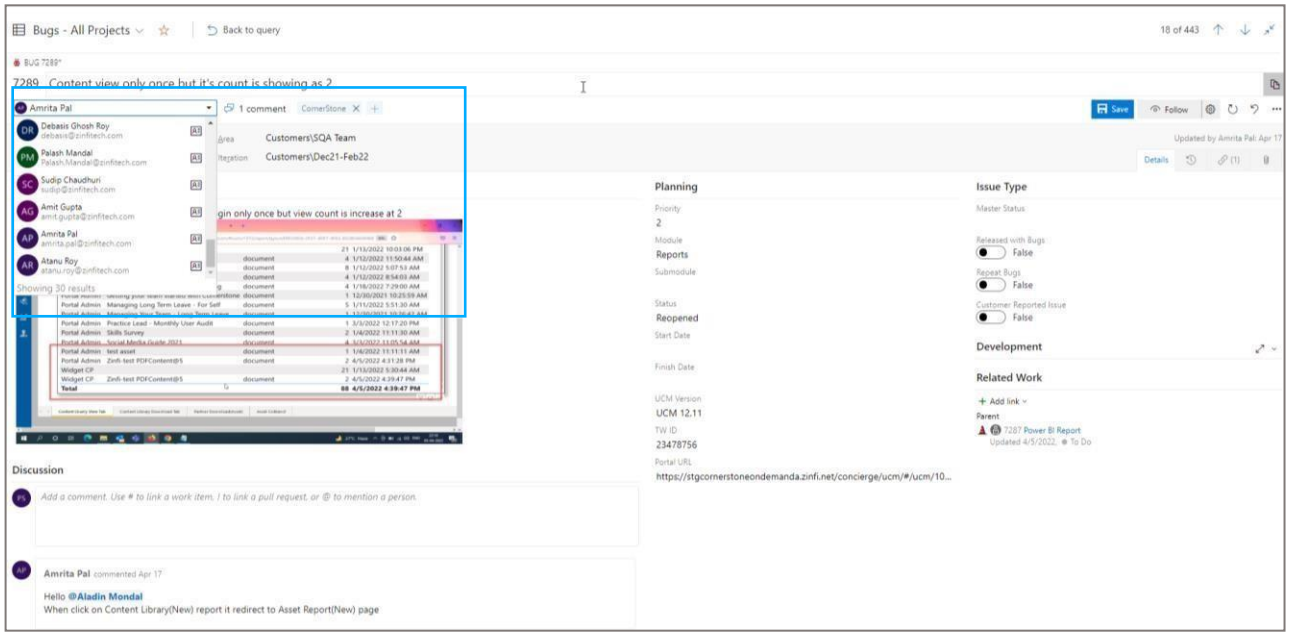
Snapshot of a reported issue, logged in Azure DevOps:



 |

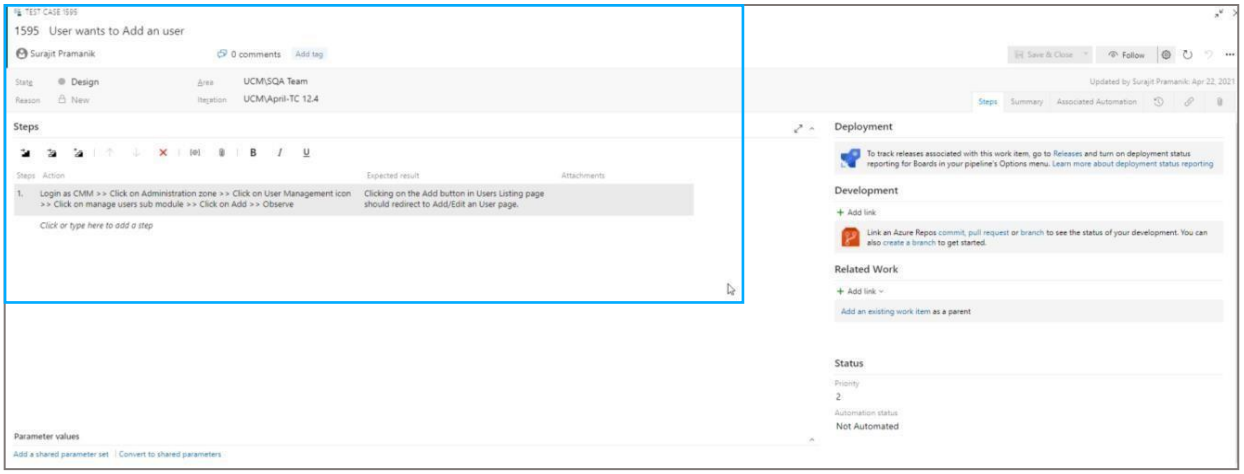
Bugs identified by SQA Team personnel are assigned to Engineering Team through Azure DevOps. The below image defines a bug identified and being assigned to the Engineering Team for code/query rectification:



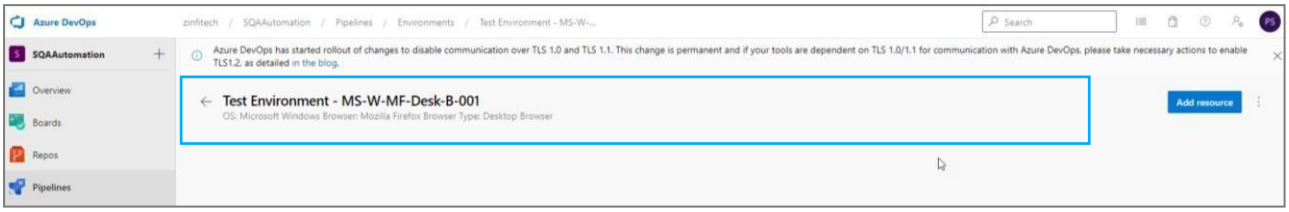


Test Case Design and Development

At this stage, our SQA Team design test cases or checklists covering requirements in Azure DevOps. Test cases outline conditions, test data (prepared at the test design stage as well), and test steps needed to validate functionality and state an expected test result. When test automation is in scope, our test automation engineers create test automation scenarios at the test design stage as well. UFT One is utilized for automation of Test Cases.

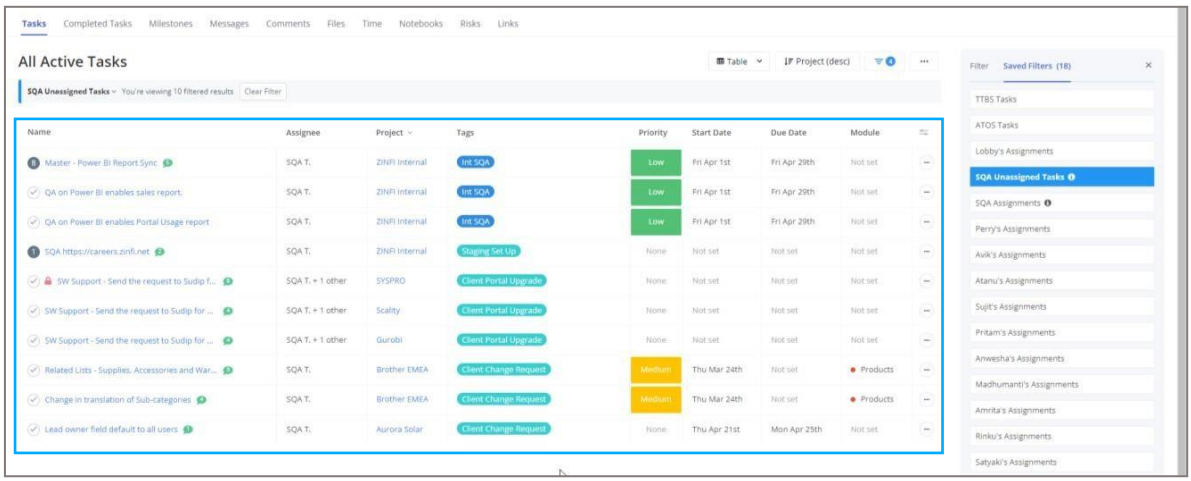


Also, the test environment is prepared for test case execution. The test environment closely mimics the production environment in terms of hardware, software and network configurations, operating system settings, available databases, and other characteristics.



We use Azure DevOps to design and store test cases. 90% of our projects are based on the standard modules of our product and we utilize the written test cases in Azure DevOps and use them as the test cases for the customer projects during execution.

Once the development team issues the release notification (containing the list of implemented features, fixed defects, known issues and limitations), the test team identifies software functionality that has been affected by the introduced changes and determines test suites required to cover the scope of the deployed build. Manual test engineers are assigned test cases and they execute the designed test cases, submitting found defects in the defect tracking system at Azure DevOps.

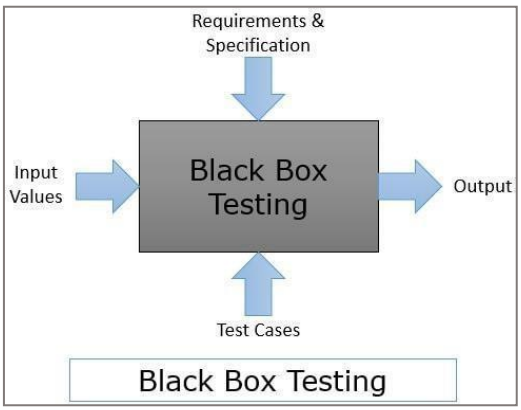


 |

Test Execution and Defect Reporting

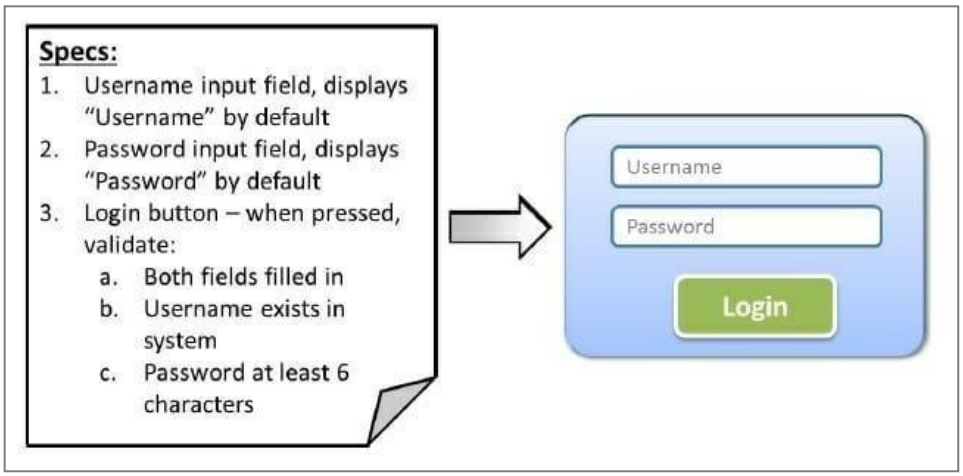
Testing is carried out module wise; in various cycles following “Black Box” Testing Techniques.

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer. In this method, tester selects a function and gives input value to examine its functionality and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Normally, after 1st cycle of testing, the bug report is generated for developer to fix the issues. Next Cycle of testing is done after the bug fixing and closure of all issues, detected during 1st cycle of testing. Test Execution and Defect Reporting is managed using Azure DevOps. *(again, see above for example of this)*

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality. It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique.



Example Black Box Test Case Specification for Portal Login

Manual test engineers execute the designed test cases, submitting found defects in the defect tracking system, while test automation engineers execute automated test scripts and generate test reports. The test team performs API and Performance Testing as well.

Once the found defects are fixed, test engineers retest functionality in question and perform regression testing to make sure that bug fixes neither broke the related functionality nor made it different from that specified in the requirements.

Test Methodologies

ZINFI SQA Team follows two test approaches depending upon the volume of release:

- Test Approach - Releases with Major/Critical Features
- Test Approach – Releases with Small Change Requests/Minor Features

Test Approach - Releases with Major/Critical Features

For big releases like initial release of customer branded ZINFI UPM portal which may include number of major critical features, we follow our standard QA procedure starting from Resource Planning, Test Case creation, Test Environment setup, Test Case execution to final acceptance process and wrap up, which are formulated below:

Step 1 - Resource Planning

For large releases that demand dedicated resource involvement for weeks of work hours, we always set a backup team to minimize risk. These backup resources take active part in all Documentation, KT (Knowledge Transfer) sessions, as well as in internal discussions so that they can readily jump into the testing process if such a situation demands.

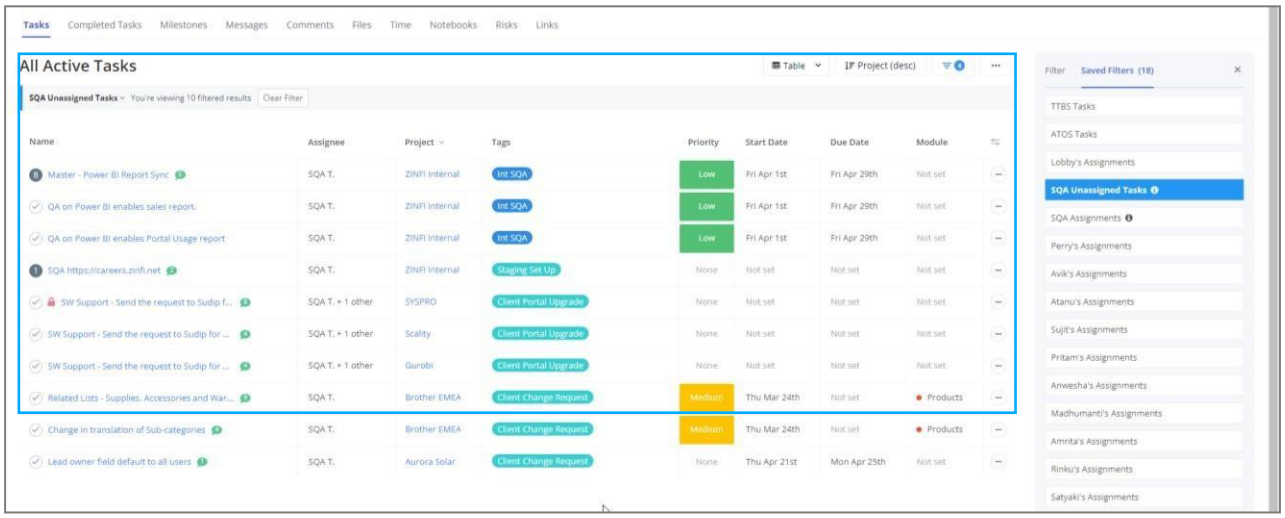
Through a combination of browser-based tools—Test plans, Progress report, Parameters, Configurations, Runs, and Test tools—and DevOps integration features, Azure Test Plans supports the following test objectives:

- Planned manual testing. Manual testing by organizing tests into test plans and test suites by designated testers and test leads.
- User acceptance testing. Testing carried out by designated user acceptance testers to verify the value delivered meets customer requirements, while reusing the test artifacts created by engineering teams.

Test cases and test suites linked to user stories, features, or requirements supports end-to-end traceability. Tests and defects are automatically linked to the requirements and builds being tested, which also helps tracking the quality of requirements. Test Engineers use the Test plans hub to define test plans and test suites.

Resource planning is choosing relevantly skilled resources/test engineers and matching them with tasks for ensuring the SQA project succeeds. What's important, resource plans may change and should be adjusted regularly to reflect changes in the scope, employees' availability, etc, so that the SQA schedule project is always up to date. To make our plans more reliable, we use historical data from past projects. Schedules, estimates, or the performance of team members we normally consider. Data from projects, maintained at Azure DevOps about delivery summary gives us a baseline for a managed resource planning process.

(Below is a view from the resource planning tool)



Step 2 - Test Case Creation

ZINFI takes considerable care in developing and managing test cases. Developing accurate, clear, and concise test cases is critical to the success of “black box”, functional testing. The first step in developing test cases is receiving and reviewing the Functional Requirements Document(s), or “FRD”, provided by the client. In case the FRD is not provided for internal releases, our Solutions Engineering and Product Teams define the Use Cases and the process flow for the operation of a specific module. These Test Cases are employed by the SQA Team. The FRD review process includes:

- Unit Level Functionality dissected
- Understanding of each pages’ rendering logic
- Creation of Application flow
- Integration Scenarios

Example FRD - Process Algorithm - Partner Onboarding

Definition

This will be the process flow for the new partners getting onboarded into the TTBS system using the ZINFI UPM platform. The partners will go through a sequence of events & activities to get this process step completed.

Actor(s)

- Partner Prospect
- Lead PDM
- PDM
- TTBS Finance Team

Sequence of Events

- The partner prospect initiated the partnership request by clicking through link on TTL website/ campaign/ Referrals.
    - INPUT: Partner Clicks on link.
    - OUTPUT: Partner lands on UPM New Partner Registration Page.

Derived Use Case

- Input Parameter: Partner Clicks on link.
- Acceptance Criteria: Partner lands on UPM New Partner Registration Page.

We always maintain a comprehensive process to design test cases by adding detailed test steps and test data so that everyone with project knowledge can execute test cases to verify actual test result or can regenerate issues with ease. We also implement Peer to Peer review to ensure that every scenario has been covered in our test cases.

A Test Case, thus is created through the following steps from the “Functional Requirements Document, or internally generated use case:

Author tests using test cases - We define manual test cases by defining the test steps and optionally the test data to reference. Test suites consist of one or more test cases. We also share test cases within test suites. The Grid view for defining test cases supports copy, paste, insert, and delete operations.

Specification-Based or Black-Box techniques are utilized - leveraging the external description of the system such as module-process specifications, and client’s requirements to design test cases. The technique enables testers to develop test cases that provide full test coverage.

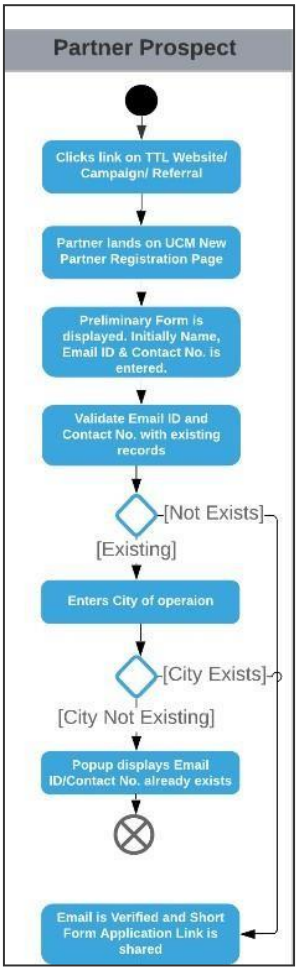
Acceptance Criteria - is looked for as the exit criteria for each Test Case. The following is an example of an Acceptance Criteria for a Test Plan.

1. The email Id & contact to be validated while form submission, duplicate record would not be allowed to proceed with error message.
2. Email address used in the form to be validated against existing PRM records, only validated email address will be allowed to proceed with form submission.
3. The 360-evaluation form to be completed and based on the score the application is to be approved & rejected.
4. The finance team validates PAN/Aadhar/GST No to be deduped in the existing database, this would be an automated process in the PRM system.
5. The finance team also matches the application/documents with Inactive partner database, this would be an automated process in the PRM system.
    a. If match found
        i. Blacklisted Partner/Debit balance exists: Application cannot be processed.
        ii. Dues clear: Application to be processed. However, there would be a vendor code already existing with the system so no new vendor code to be created.

6. Partner pays the security deposit and share the transaction ID which is verified by the TTBS finance team.
7. The account created in ZINFI UPM, and the CP is mapped with the PM.
8. The CP receives the Welcome mail sent to Partner on his personal mail ID with:
    a. PRM/Outlook login credentials
    b. PM/PDM/ZH hierarchy
    c. Pay-out Policy attachment
    d. Induction link
9. All email notifications are triggered at different stages with correct content and to the right persons.

Process Flow based on Client requirements – Based on the FRD or Module-process flow as defined by the Client or the Product/Solutions Engineering Team we determine what to Test. These provides us inputs to define the steps in the test process that will satisfy testing for this use case, and justification to picking those steps.

Example Process Flow from an FRD:



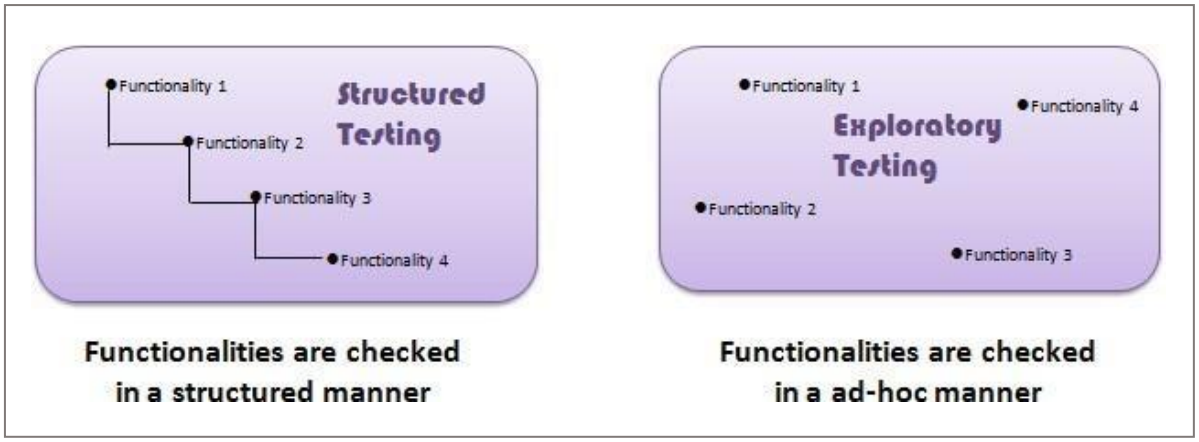
Step 3 - Test Approach, Methods & Tools

Big releases or critical features involve QA in multiple environments or platforms before releasing it in the Production environment. We have SIT and UAT environments to ensure that the feature under test undergoes utmost sanity testing so that when it is pushed into Production, it appears as almost bug free.

We follow manual black box testing techniques to test features. We involve such approaches as:

1. Exploratory Testing

   Exploratory Testing is a type of software testing where Test cases are not created in advance, but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity. Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.



2. Use Case-Based Testing to identify issues.

   Use case testing is a technique that helps to identify test cases that cover the entire system, on a transaction-by-transaction basis, from start to finish. It is a description of a particular use of the system by a user.

   Sequence of Events

   The partner prospect initiated the partnership request by clicking through link on TTL website/ campaign/ Referrals.

   INPUT: Partner Clicks on link.

   OUTPUT: Partner lands on UPM New Partner Registration Page.

   Derived Use Case

   Input Parameter: Partner Clicks on link.

   Acceptance Criteria: Partner lands on UPM New Partner Registration Page.

3. For final UAT phase, we normally have collaborative sessions with client which can be considered as Acceptance Testing. Platform walkthrough based on business requirements and acceptance criteria are conducted. Test Engineering Teams are involved with PSM Team and Customer– for the

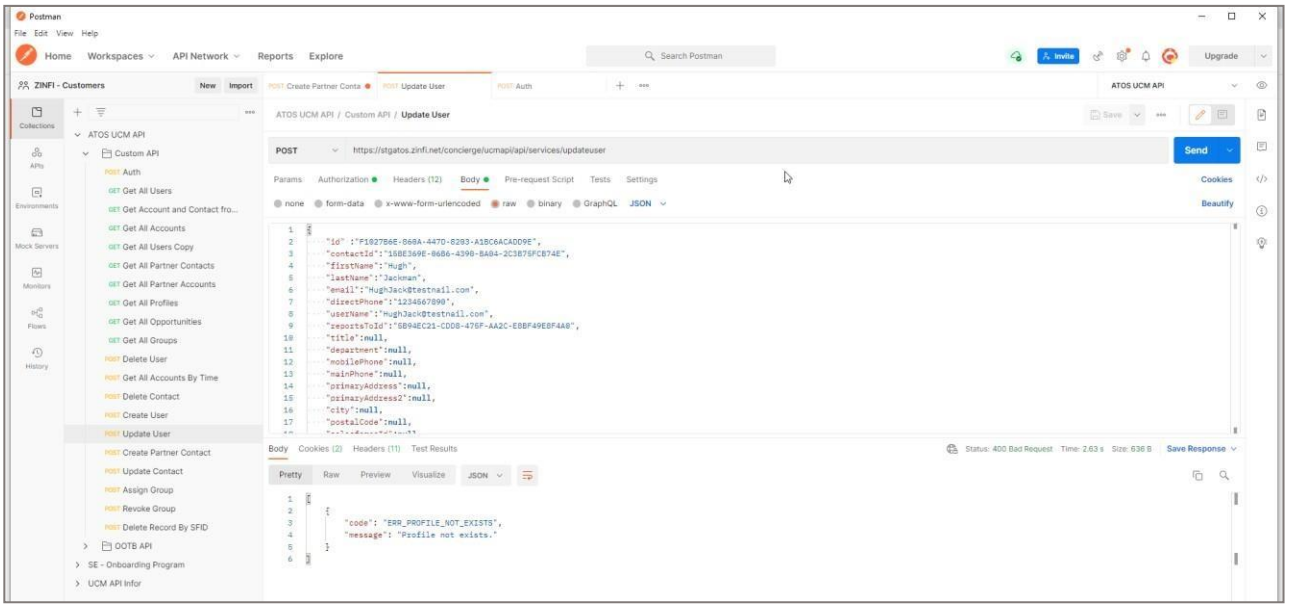
walkthrough session. User acceptance testing (UAT) helps ensure our test teams deliver the value requested by customers. We also create UAT test plans and suites and invite the PSM team along with a customer representative to execute these tests and monitor test progress and results through Azure DevOps.

Step 4 - APIs & Performance Testing

To Test APIs, we use Postman.

Postman is an API platform for building and testing APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration for us to create better APIs—faster. The Postman platform includes a comprehensive set of tools that help accelerate the API lifecycle—from design, testing, documentation, and mocking to the sharing and discoverability of our APIs. The Postman API client is the foundational tool of Postman, and it enables you to easily explore, debug, and test your APIs while also enabling you to define complex API requests for HTTP, REST, SOAP, GraphQL, and WebSockets. The API client automatically detects the language of the response, links, and format text inside the body to make inspection easy. The client also includes built-in support for authentication protocols like OAuth 1.2/2.0, AWS Signature, Hawk, and many more. Through the API client, we organize requests into Postman Collections to help us organize our requests for reuse.

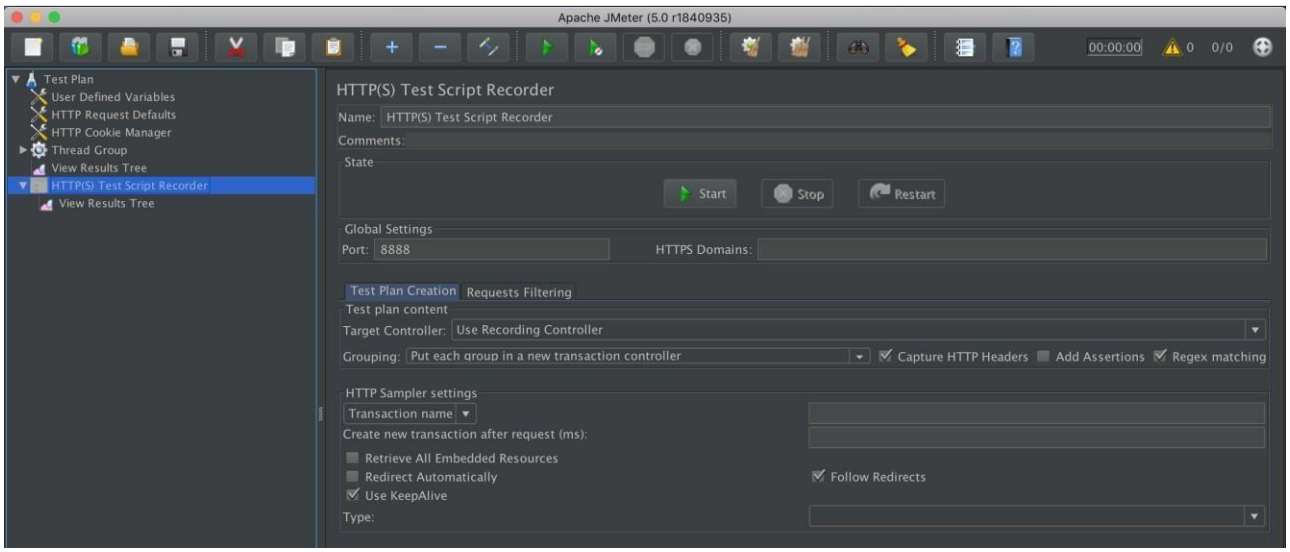
We can build and run tests directly in Postman - Collection Runner that enables us to run and test a Postman Collection directly from the command line. Postman can be used to write functional tests, integration tests, regression tests, and more.



 |

In large and critical releases, we use Apache JMeter 5.2 to run performance testing.

The Apache JMeter™ application is open-source software, designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. Apache JMeter is used to test performance both on static and dynamic resources, Web dynamic applications. It is also used to simulate a heavy load on the server, network, or object to test its strength or to analyze overall performance under different load types.



Step 5 - Exit Criteria

For large releases, the following standard exit criteria is followed by us:

- Business Requirements got fulfilled.
- Execution of all Test Cases.
- Desired and sufficient coverage of the Requirements and Functionalities under the Test.
- All the Identified Defects are Corrected and Closed.
- No High Priority or Severity or Critical Bug has been left out.

Example Acceptance Criteria:

Sequence of Events

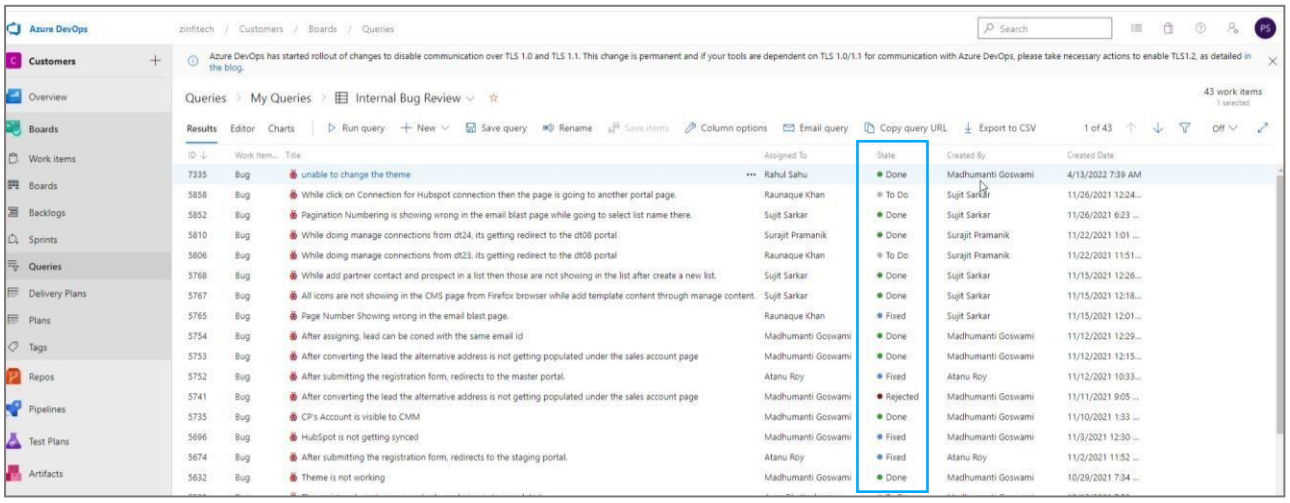
- The partner prospect initiated the partnership request by clicking through link on TTL website/ campaign/ Referrals.
    - INPUT: Partner Clicks on link.
    - OUTPUT: Partner lands on UPM New Partner Registration Page.

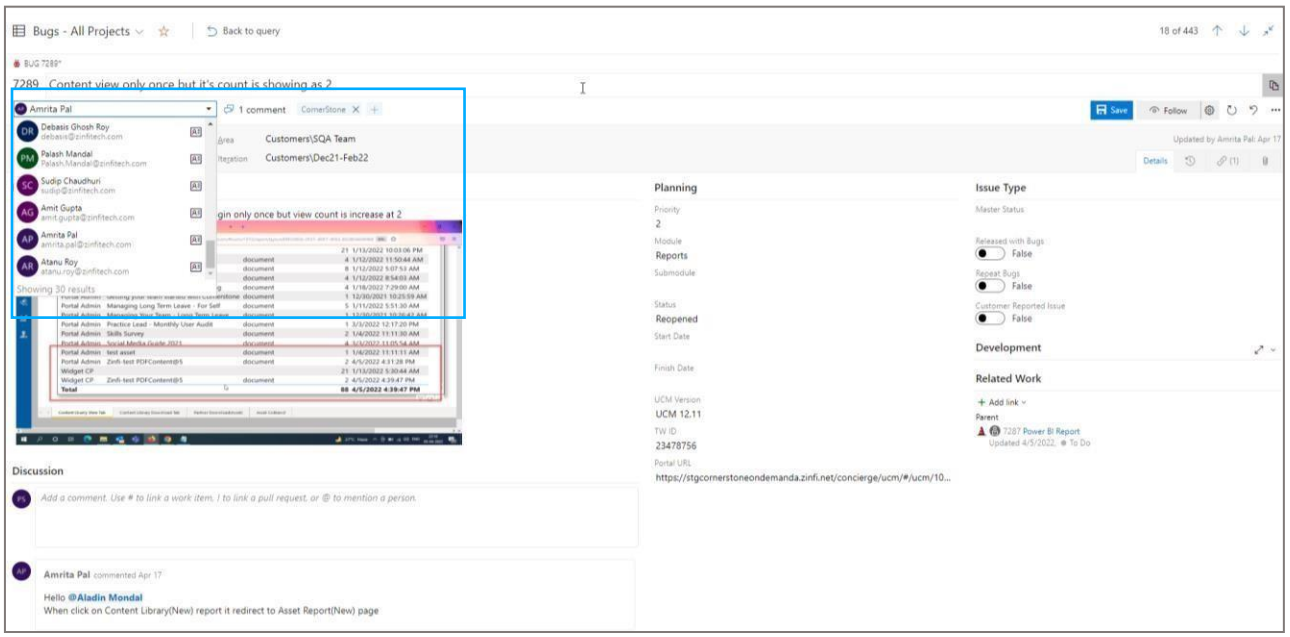
 |

Derived Use Case

- Input Parameter: Partner Clicks on link.
- **Acceptance Criteria: Partner lands on UPM New Partner Registration Page.**

A successful Test Case State is updated as Done/Fixed. A failed Test Case State is updated as Failed and assigned to Engineering Team with comments. Once, the Bug is fixed, the Engineering team assigns it back to the Test Engineer. The Test Engineer re-tests the Test Case and either updates it as Fixed or Re-opens it- if not fixed yet. Once all test cases State are updated to Done then the SQA Team considers the test plan as Successful and passes on for Client Acceptance, by issuing a Success Ticket to the PSM Team.





 |

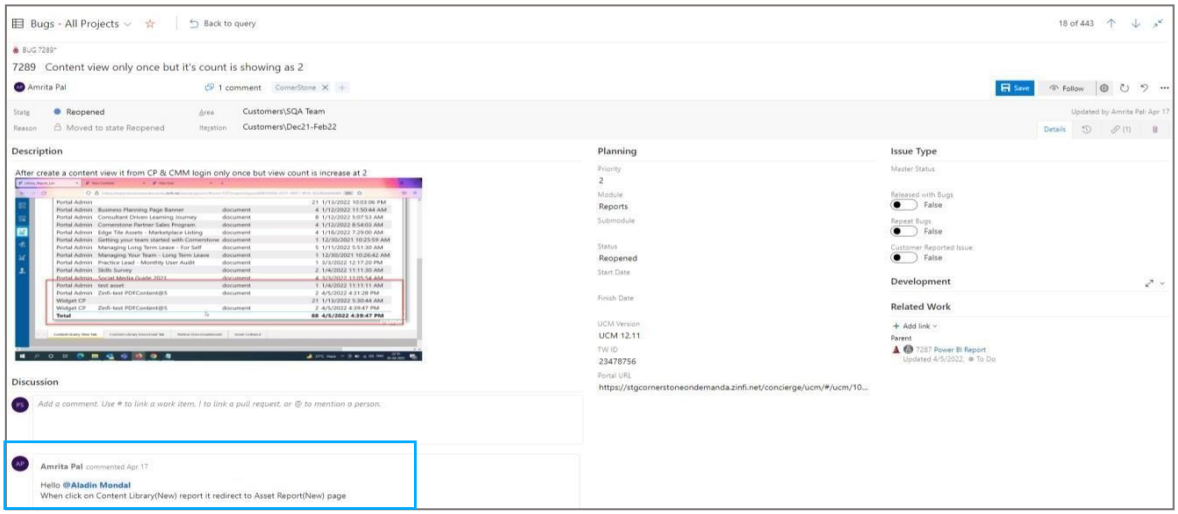
Test Approach – Releases with Small Change Requests/Minor Features

For small, minor feature releases, we follow a smaller approach to our standard QA procedure:

We review the SRS and supplementary documents.

1. A Test Case, thus is created through the following steps from the Functional Requirements Document/SRS, or internally generated use case:
2. Author tests using test cases - We define manual test cases by defining the test steps and optionally the test data to reference. Test suites consist of one or more test cases. We also share test cases within test suites. The Grid view for defining test cases supports copy, paste, insert, and delete operations.
3. Specification-Based or Black-Box techniques are utilized - leveraging the external description of the system such as module-process specifications, and client's requirements to design test cases. The technique enables testers to develop test cases that provide full test coverage.
4. Acceptance Criteria - is looked for as the exit criteria for each Test Case.

Execute tests on the feature - conducted on an Ad Hoc basis. We identify issues and log those in Azure DevOps and convey it to the responsible developer through Teamwork Tasks.



Performance Test

Normally, we do not run Performance tests on minor feature release or on small change requests.

Exit Criteria

For minor releases, we follow the below-formulated standard exit criteria

- Business Requirements got fulfilled.
- All the identified defects are corrected and closed.
- No high priority or severity or critical bug has been left out.

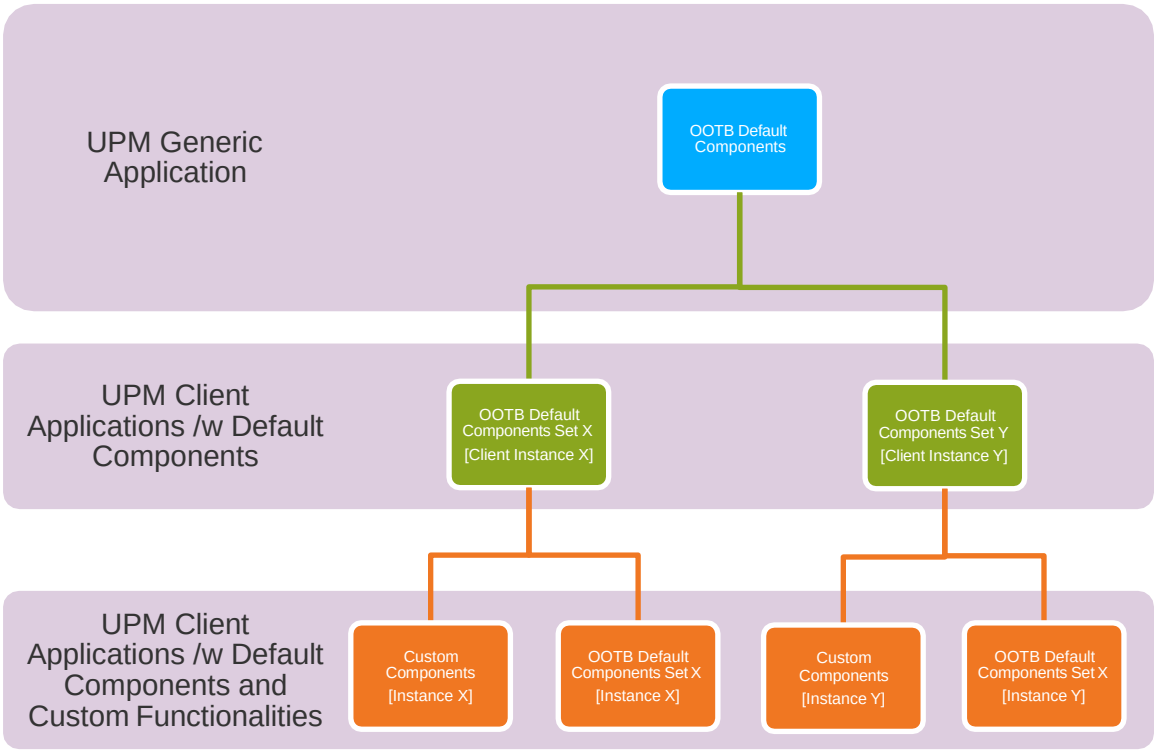
Test Automation Components

OOTB UPM Components

- OOTB UPM Default Components are generic and provided to all Clients/OEMs inclusive of OOTB Modules with UPM OOTB process flows.
- QA will be conducted on the default components from a centralized / generalized perspective and then deployed to different client instances. *Example: Generic Partner Onboarding Process Flow of UPM Partner Onboarding Flow.*
- QA time factor would be reduced from n (no. of UPM client instances) -> 1 for the OOTB components deployed, post replication. *Therefore, for all Client Portals with the Generic Partner Onboarding Flow SQA Activities would be conducted once to ensure that feature is good to go across all client platforms.*

Custom Feature - Components and Functionalities

- Custom Components and Functionalities are specific to the Client/OEM UPM instance.
- QA process will be conducted manually for each specific Client/OEM instance – Custom Components.



 |