# Utility Tool - Master Test Plan

Please Note: Utility is a Tool, not an application and may not need full testing as required by usual applications.  Master test plan below is based on requirements for full applications and may change for this Tool.

# 1.    Introduction

This Master Test Plan details a systematic approach to testing the Utility Tool. The document outlines the test objectives, roles and responsibilities of individuals involved with testing activities, test inclusions and exclusions, and the overall testing approach. It outlines the different types of testing that will be performed to test the product. The document also describes the tools that will be used to create automated test scripts and metrics, test deliverables, schedule, and test environments.

## 1.1.   Purpose

The purpose of this Master Test Plan is to define a common understanding on the test approach and process that will be followed by the Teams developing this project.

## 1.2.     Test Objectives

This Master Test Plan supports the following objectives:

* Identify the types of tests to be performed to validate product quality and production readiness
* Identify the testing roles and responsibilities
* Identify the tools to automate majority of the test cases.
* Identify the test process to deliver a bug free product for User Acceptance testing at the end of each Release Increment
* Identify the test process to deliver a bug free demonstrable product to Pre-Prod and Production environments at the end of each Release Increment (PSI)
* Identify and define the rules to maintain a consistent test environment to execute the different types of testing
* Define and manage Test Data that supports both the manual and automated test runs
* Identify any potential risks that might impede testing of the App
* Identify the process for creating and maintaining the Requirements Traceability Matrix (RTM)
* Provide links to artifacts related to testing.

## 1.3.    Roles and Responsibilities

###### Table 1: Roles and Descriptions

| **Role** | **Description** |
| --- | --- |
| Delivery (development) Team | Persons that build, unit test, and integration test the product/product component. |
| Program Manager | Person that has overall responsibility for the successful planning and execution of a project; person responsible for creating the Master Test Plan in collaboration with the Development Manager. |
| System Team | Persons responsible for creating and maintaining the Continuous Integration (CI) environments, including assisting developers in creating the necessary development workspace on their local machines. Also responsible for setting up needed deployment planning and scripting for both integrated testing and production environments with necessary security and certification requirements addressed and regularly maintained. Targets product delivery, quality testing, feature development, and maintenance releases in order to improve reliability and security and faster development and deployment cycles. |
| Stakeholder | Person that hold a stake in a situation in which they may affect or be affected by the outcome. They are responsible in mitigating or fixing any impediments that may arise throughout the project life cycle. (Examples – identifying V&V team members, UAT team members, access to contractors to Pre-Prod or UAT environments, approval of requirements, etc) |
| Test Lead | An experienced Test Analyst or member of the System Team that leads and coordinates activities related to all aspects of testing based on an approved Master Test Plan and schedule. Responsible for collecting the necessary test metrics and responsible for making sure a quality product is delivered to the Customer. |
| Test team | Persons that are part of the Delivery or System Team responsible in creating and executing tests according to the acceptance criteria of a user story. They are responsible in creating the test infrastructure and providing requirements to build the necessary test environments.  In addition, they also create the test data that is required to validate the requirements. At times they might be responsible in assisting UAT and V&V teams in UAT and V&V testing. |
| User Acceptance Test team (UAT) | Persons from end user community who help in testing the product in a pre-prod environment/UAT environment (in MAE current environment is SQA). |
| Verification and Validation (V&V) Testers | The V&V team sees applications from the transition from Software Quality Assurance (SQA) and development completion to the V&V Test Intake process, which includes a Functional Findings Review to verify that the application is ready to test; through Test Execution and Analysis, which includes Functional, 508, and Security testing; to Test Summary Reporting that delivers all documentation of test analysis and results so that the application may proceed into final compliance review, Pilot/Initial Operating Capability (IOC), and National Release or go back for a development remediation cycle |
| Compliance Testers | The purpose of the VA compliance reviews throughout the mobile app lifecycle is to ensure that applications meet the standards of the VA. This includes accessibility, safety, privacy and security standards, and more. There are a total of 10 compliance bodies that review apps. |
| Product Owner | The Product Owner represents the stakeholders and is the voice of the customer.  He/She is accountable for ensuring that the team delivers value to the business. The Product Owner writes (or has the team write) customer-centric items (typically user stories), ranks and prioritizes them, and adds them to the product backlog. |

## 1.4.    Processes and References

The processes that guide the implementation of this Master Test Plan are:

* Requirements Traceability Matrix (RTM)
* Defect Metrics
* Definition of Done
* Test Preparation
* Product Build
* Independent Test and Evaluation

The references that support the implementation of this Master Test Plan are:

* [Section 508 Office Web Page](http://DNS/508workgroup)
* [Privacy Impact Assessment - Privacy Service](http://dns/DNS.asp)
* [Compliance SOP](file:///C:\display\COM\Compliance+SOP)

# 2.    Items To Be Tested

The following is what will be tested and validated.

|  |  |  |
| --- | --- | --- |
| **Test Item** | **Type** | **Version** |
| Utility Tool | Code | 1.0.0 |

2.1.          Overview of Test Inclusions

The following components and features and combinations of components and features will be tested:

**Utility Tool**

1. SA can authenticate with VAMF Admin credentials
2. SA can select VistA instances they are assigned to for updating including multi-divisional VistA instances to edit/update
3. Ability to update information in the system by VistA instance
   1. Customized messages if user is not registered
   2. Customized messages if direct appointment scheduling not available
   3. Parameters for length of time to search historical appointments
   4. Clinic contact information

2.2.   Overview of Test Exclusions

The following components and features and combinations of components and features will not be tested:

All components, features, and combination of components/features that are delivered will be tested at this time.

# 3.    Test Approach

There are three levels of testing that comprise the Testing Approach: Unit Testing, Integration Testing, and System Testing.  All three levels of testing include test scripts that are executed in an automated fashion. Specific tools are used to automate the tests at each level. All three levels of testing will be performed based on the requirements of the User Story and the defined acceptance criteria. In order for a User Story to meet the acceptance criteria, every User Story must have completed at least one level of test coverage. The only exception to this is stories that are not testable (Example: User Story that deals with documentation.).

## 3.1.    Product Component Test

All components of the applications will be thoroughly unit tested by the Development Team. Developers are responsible for creating unit tests for the components that they build. A User story includes tasks for unit test creation and unit testing activities. Developers are responsible for monitoring the unit test build after the code and related unit tests are checked in. If a developer’s code check-in breaks a build, the developer must notify the delivery team(s) that the fix is underway to address the build server breakage.  If build server failure cannot be resolved within 15 minutes, the developer must take action to revert the code and get the build server running green again. Care must be taken not to check-in on a broken build.

**JUnit**will be used to develop and execute unit tests to validate the backend.

**Jasmine**will be used to develop and execute unit tests to validate the frontend.

## 3.2.   Component Integration Test

The purpose of the Product Integration Testing is to expose the defects in the interfaces and interaction between integrated components as well as verifying installation instructions.

Developers are responsible for creating the integration tests for a user story, as appropriate in support of the user story acceptance criteria. The User Story must include tasks for creating and executing Integration tests. As part of the Code Review, integration tests must also be reviewed for proper coverage. Developers are responsible for monitoring the integration test builds after the code and related integration tests are checked in. If a check-in breaks a build, the responsible developer must notify the development team that the fix is on the way or action must be taken to revert the code if a quick fix cannot be checked in. It is advisable not to check into a red build.

Integration Test tool – JUnit will be used in writing the integration tests.

Below are some of the scenarios that will be validated as part of Integration testing?

-       Test data saved correctly in Mongo DB

-       Test data displays correctly in the UI

-       Retrieving data from the rest calls.

-       Updating Mock services without interaction of the front end.

-       Resource integration to test end points

## 3.3.   System Tests

Delivery Team Developers and Testers are responsible for creating the system tests. System tests are tests that test the entire system as one entity across all functional feature components. End-to-End testing is validated by system tests. Testers work with developers, architects, and UAT Testers and V&V personnel to define the tests executed under system testing. System tests include functional, non-functional, performance, and security testing. System test includes end-to-end integration testing of interfaces between the front-end and backend calls.

* UI Testing verifies the user interface is working as expected and meeting the criteria defined in requirement documents.
* Smoke testing is performed to ensure that system is stable enough and ready for test analyst to perform system testing.
* Automated Regression Testing is performed with each enhancements/bug fixes that an existing functionalities are working as expected.
* Complete Regression testing is performed at the completion of increment development.
* Accessibility Testing is a verification testing for users with disabilities.

The components and functionality listed in Section 2.1 will be tested during system testing.

All defects found during System Testing are reported and documented as a JIRA Bug Issue Type in the project related to the app.

## 3.4   Functional Tests

Testers from each scrum team perform functional testing. Testers will work with Analysts and Product Owners to understand the acceptance criteria defined for each user story. Testers will write test steps to validate the acceptance criteria. User Stories will have tasks defined for Quality Assurance (QA) activities. All tests are checked in and must be passing on the build server at the end of a Sprint (meeting the Definition of Done). User Story must not be closed if the acceptance tests are not passing.

Test Data – The testers identify Test data required for testing the acceptance criteria and with help from the development team the test data are created and staged both in the local dev environments and other environments like the demo and CI environments that execute the acceptance tests.

Clear documentation of the test data used is maintained on the wiki.

Test Environment – Acceptance test creation and validation are performed in local dev environment. Acceptance tests are tagged to be run

Test Tools –

Functional Test Tool – **Selenium/Watir**will be used to develop and execute functional tests.

**Test Data** – Please refer to this wiki page for the test data requirements

**Testing Tools**

|  |  |
| --- | --- |
| **Tool** | **Description** |
| Selenium/Watir | Automated Test Tool |
| Devices for Manual Testing | iPhone, Android (small device)  Windows 7 laptop (larger device) |
| Voice Over | Section 508 Testing |

## 3.4.2   User Functionality Test

The VA Stakeholder will conduct User Functionality Testing in the MAE Demo environment or SQA environment.  All issues reported from the stakeholders will be entered in Jira.

## 3.5.    Enterprise System Engineering Testing

The VA Project Team will assist in triaging any issues found during Enterprise System Engineering (ESE) Testing of apps to development team.  The ESE compliance body will be conducting the testing.  Issues that are determined to be valid will added to the JIRA backlog to be fixed.

## 3.6.   Initial Operating Capability Evaluation

The Development Team will conduct Initial Operating Capability (IOC) Evaluation per the contract requirements with support from the VA Project Team.  At the conclusion of the IOC testing, the Development Team will correct any defects identified as required for National Release.

# 4.   Testing Techniques

## 4.1   Build Verification Testing

Also known as Build Acceptance Test, is a set of tests run on each new build of a [product](http://en.wikipedia.org/wiki/Product_(business)) to verify that the build is stable before it is made available to the rest of the development team to deploy to their local environments and also to deploy to external environments like the demo environment.

This testing is automatically performed by various build jobs that are automatically triggered when code is committed. Some of the build jobs that validate the code checked in are the unit-test build, integration-test, smoke test and the acceptance-test.

## 4.2   Integration Testing

Integration tests are created by Developers to test the backend code. The test also validates the backend rest calls without accessing the front-end code. Integration tests are written to validate that certain functionality of the app correctly updated the database.

Tools – JUnit is created the integration tests

## 4.3    Enterprise Testing

The VA Compliance groups will complete testing and reviews to cover the Enterprise requirements. This will be conducted in parallel or after V&V testing.

**4.3.1.         Security Testing**

The Security Assessment Team will verify that all security requirements have been met as part of their review of the Utility Tool.

**4.3.2.         Privacy Testing**

The Privacy Office will complete a review of the application to ensure that veteran and employee data are adequately protected and the apps comply with the Privacy and Security Rule provisions of the Health Insurance Portability and Accountability Act (HIPAA).

**4.3.3.         Section 508 Compliance Testing**

The Development Team will ensure that the product functionality is usable from the keyboard by completing 508 testing as part of the manual testing process.  The Section 508 Program Office is responsible for performing independent compliance testing with assistive technology.  508 testing is also performed on devices like Android and iPhone using voice over.

### 4.3.4.    Multi-Divisional Testing

Multi-divisional testing is not applicable for Utility Tool.

## 4.4.    Risk-based Testing

Risk-based testing is a technique for prioritizing testing based on testing the highest risk items first and continuing to test down the risk prioritization ladder as the testing schedule permits.

The following is the link that logs the risk items for Utility Tool. [Risk Log](https://DNS/display/ARA/Risk+Log)

## 4.5.    Test Types

Test types are a group of test activities aimed at testing a component or system regarding one or more interrelated quality attributes. A test type is focused on a specific test objective, i.e., reliability test, usability test, regression test etc., and may take place on one or more test levels or test phases. Please see the test types to be performed and the party responsible for performing the test.

###### Table 2: Test Types

| Test Types | Party Responsible |
| --- | --- |
| Access control testing | Development Team  VA SMEs  V&V |
| Compliance testing | VA Compliance Groups |
| Component integration testing | Development Team |
| Installation testing | Development Team |
| Integration testing | Development Team |
| Privacy testing | VA Compliance Groups |
| Product component testing | Development Team |
| Regression test | Development Team |
| Risk based testing | Development Team  V&V  VA SMEs |
| Section 508 compliance testing | Development Team   VA Section 508 Team |
| Security testing | VA Compliance Groups |
| Smoke testing | Development Team (Integration Environment Install) |
| Usability testing | VA Human Factors Team |
| User Functionality Testing | Development Team  VA SMEs  V&V |
| User interface testing | VA Human Factors Team |

## 4.6.        Productivity and Support Tools

Table 3 describes the tools that will be employed to support this Master Test Plan.

###### Table 3: Tool Category or Types

| Tool Category or Type | Tool Brand Name | Vendor or In-house | Version |
| --- | --- | --- | --- |
| Test Management | Atlassian JIRA | Vendor |  |
| Defect Tracking | Atlassian JIRA | Vendor |  |
| Test Coverage Monitor or Profiler | N/A | N/A | N/A |
| Project Management | Atlassian/Wiki | Vendor |  |
| Performance Testing | Jmeter | In-House | N/A |
| Configuration Management | Atlassian/Wiki  Stash | Vendor |  |
| DBMS tools | MongoDB  Oracle | Vendor |  |
| Functional Test Automation | Selenium/Watir  Jenkins | Vendor |  |
| Unit test/Integration test | Jasmine/Junit | In-House | N/A |

# 5.           Test Criteria

## 5.1.          Process Reviews

The Master Test Plan under goes two reviews:

* Peer Review – upon completion of the Master Test Plan
* Formal Review – after the Development Manager approves the Master Test Plan

## 5.2.          Pass/Fail Criteria

Pass/Fail criteria are decision rules used to determine whether a test item (function) or feature has passed or failed a test.

|  |  |
| --- | --- |
| **Team** | **Test Item Pass/Fail Criteria** |
| Development Team | * Meets User Story Acceptance Criteria * Manual Test Script for User Story - All Steps Pass |
| VA SMEs | * Customer Requirements Met |

## 5.3.          Suspension and Resumption Criteria

Suspension Criteria are the criteria used to (temporarily) stop all or a portion of the testing activities on the test items. Resumption Criteria are the testing activities that must be repeated when testing is re-started after a suspension.  This will apply to the IOC testing that occurs in the production environment.

Suspension Criteria

1.       Allowing patients to create requests or appointments even if they are not registered in a facility.

2.       System allowing over booking of appointments.

3.       Booking data not retrieved from external source for some reason.

4.       Retrieving data from VistA takes longer than the accepted time frame.

**Resumption Criteria**

Defect is logged for the issue and after the defect has been resolved and verified all tests or acceptance criteria referenced in the defect must be rerun to make sure no new issues have been introduced. Rerun tests that are related to the fix or features that might be affected by the fix.

## 5.4.          Acceptance Criteria

Acceptance criteria are decision rules that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity.

* All tests are running green on the CI server
* All tests meet the acceptance criteria of the story/feature
* All 508 testing is complete and results documented and accepted by VA
* Performance testing has been completed and results documented and accepted by VA
* All critical bugs have been resolved and validated
* A feature has been validated by V&V and UAT teams
* Sign off by end user on the feature delivered

The criteria below determines if the application satisfies the acceptable level of quality:

|  |  |
| --- | --- |
| **Testing Activity** | **Acceptance Criteria** |
| V&V/Compliance Reviews | * Approval to move forward by V&V/Compliance Teams to UAT or IOC * Severity 1 and 2 Defects (Agreed upon by with VA SMEs) Fixed in Remediation Cycle 1 |
| User Acceptance Testing (UAT) | * Approval to move forward by VA SMEs to IOC * Severity 1 and 2 Defects (Agreed upon by with VA SMEs) Fixed in Remediation Cycle 1 |
| Limited Field Testing (IOC) | * Approval by VA to roll out app to additional sites for full production release * Severity 1 and 2 Defects identified during IOC (Agreed upon by with VA SMEs) Fixed before full production release |

# 6.    Test Deliverables

Table 4 lists the test deliverables for the CMS project.

###### Table 4: Test Deliverables

| **Test Deliverables** | **Responsible Party** |
| --- | --- |
| Master Test Plan | Development Team |
| Test Execution Risks (Part of Master Test Plan) | Development Team |
| Test Schedule | Project Manager(s)  V&V/Compliance Teams |
| Test Cases/Test Scripts | Development Team  V&V/Compliance Teams |
| Test Data | Test Data Team |
| Test Environment | MIS Team |
| Test Evaluation Summaries | Development Team |
| Traceability Report or Matrix | Development Team |

## 7.    Test Schedule

The test schedule is set and maintained by the VA and other groups performing testing activities.  List the major testing milestones. When appropriate, reference other workflow documentation or tools, such as the Project Management Plan, or Work Breakdown Structure (WBS). Put a minimum amount of process and planning information within the Master Test Plan in order to facilitate ongoing maintenance of the test schedule.

Test Schedule: TBD

###### Table 5: Testing Milestones

| Testing Milestones | Responsible Party | Dates |
| --- | --- | --- |
| Development/Pre-V&V | Development Team  MIS Team  V&V/Compliance Teams/Stakeholders | TBD |
| V&V | V&V | TBD |
| UAT | Development Teams  UAT | TBD |
| Final Compliance | V&V/Compliance Teams | TBD |
| IOC | Development Team  Evaluation Sites | TBD |

# 8.    Test Environments

A test environment is an environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

Stakeholders will perform testing in the Demo environment during development.

V&V and Compliance groups will perform testing in the SQA environment for their testing activities.

VA SMEs with assistance from Development Team will conduct UAT in the Demo or SQA Environment.

IOC testing will be conducted in the Production Environment to a limited number of sites.

## 8.1.    Test Environment Configurations

Successful testing requires control of the test environment. Unplanned changes to the test environment may introduce new defects, alter the expected test results, and thus invalidate the test cases. Successful testing requires controlled access to the test environment, an environment that replicates the production environment as closely as possible.

Any changes to the SQA, Pre-production and Production environments are controlled by the CCB.  The installation and any updates to the Utility Tool in these environments will require tickets opened in JIRA and approval from the CCB.

## 8.2.     Base System Hardware

The test system (SQA) will simulate the production environment as closely as possible, scaling down the concurrent access and database size, and so forth, if and where appropriate.  The SQA environment is maintained and managed by the MIS team.

## 8.3.     Base Software Elements in the Test Environments

Table 7 describes the base software elements that are required in the test environment for this Master Test Plan.

###### Table 7: Software Elements

| Software Element Name | Version | Type and Other Notes |
| --- | --- | --- |
| Windows 7 | 7 | Operating System |
| Internet Explorer | 11 | Internet Browser |
| Firefox | 31 | Internet Browser |
| Safari | 7 | Internet Browser |
| Android |  | IOS version |
| iPhone | 8 + | IOS version |
| NVDA | Latest | VO on windows |
| VO | Built in | VO for iPhone |
| Talkback | Built in | VO for Android |

# 9.     Staffing and Training Needs

Table 8 describes the personnel resources needed to plan, prepare, and execute this Master Test Plan.

###### Table 8: Staffing Resources

| Testing Task | Quantity of Personnel Needed | Test Process | Duration/ Days |
| --- | --- | --- | --- |
| Create the Master Test Plan | 2 | Test Preparation | 3 days |
| Establish the Development Test Environment | 1 | Test Preparation | 2 days |
| Perform System Tests | 1-2 | Product Build | 1 day |
| UAT | 10 | UAT | 10 days |
| IOC | TBD | IOC | 60 days |

There are no training options required at this time for providing necessary skills to execute the test plan.

Training Needs

| Name | Training Need | Training Option | Estimated Training Hours |
| --- | --- | --- | --- |
| N/A | N/A | N/A | N/A |

# 10.   Risks and Constraints

Risks are documented and managed on the wiki [Risk Log](https://DNS/display/ARA/Risk+Log)

# 11.   Test Metrics

Metrics are a system of parameters or methods for quantitative and periodic assessment of a process that is to be measured.

Test metrics may include, but are not limited to:

* Number of test cases (pass/fail)
* Percentage of test cases executed
* Number of requirements and percentage tested
* Percentage of test cases resulting in defect detection
* Number of defects attributed to test case/test script creation
* Percentage of defects identified; listed by cause and severity

The Development Team will work with the VA to determine which test metrics will be applicable for the UAT Testing and documented as part of the Test Summary.

Attachment A - Approval Signatures

The Master Test Plan documents the project’s overall approach to testing and includes:

* Items to be tested
* Test strategy
* Test criteria
* Test deliverables
* Test schedule
* Test environments
* Staffing and training needs
* Risks and constraints
* Test Metrics

This section is used to document the approval of the Master Test Plan during the Formal Review.  The review should be ideally conducted face to face where signatures can be obtained ‘live’ during the review however the following forms of approval are acceptable:   
  
1.  Physical signatures obtained face to face or via fax   
2.  Digital signatures tied cryptographically to the signer   
3.  /es/ in the signature block provided that a separate digitally signed e-mail indicating the signer’s approval is provided and kept with the document.

NOTE:  Delete the entire section above prior to final submission.

REVIEW DATE: <date>

< Program/Project Manager >

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed:                                                                                                             Date:

< Business Sponsor Representative >

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed:                                                                                                             Date:

< Integrated Project Team (IPT) chair >

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed:                                                                                                             Date:

< Enterprise Systems Engineering (ESE) Representative >

A. Test Type Definitions

Test analysts use “test types” to validate the system or application under test. This table presents a listing of possible test types and their definitions that may be utilized during the Product Build, Independent Testing, Operational Readiness Review (ORR) and Initial Operating Capability (IOC) Testing.

| Test Type | Definition |
| --- | --- |
| Access Control Testing | A type of testing that attests that the target-of-test data (or systems) are accessible only to those actors for which they are intended, as defined by use cases. Access Control Testing verifies that access to the system is controlled and that unwanted or unauthorized access is prohibited. This test is implemented and executed on various targets-of-test. |
| Benchmark Testing: | A type of performance testing that compares the performance of new or unknown functionality to a known reference standard (e.g., existing software or measurements). For example, benchmark testing may compare the performance of current systems with the performance of the Linux/Oracle system. |
| Build Verification Testing  (Prerequisite: Smoke Test) | A type of testing performed for each new build, comparing the baseline with the actual object properties in the current build. The output from this test indicates what object properties have changed or don't meet the requirements. Together with the Smoke test, the Build Verification test may be utilized by projects to determine if additional functional testing is appropriate for a given build or if a build is ready for production. |
| Business Cycle Testing | A type of testing that focuses upon activities and transactions performed end to end over time. This test type executes the functionality associated with a period of time (e.g., one-week, month, or year). These tests include all daily, weekly, and monthly cycles, and events that are date-sensitive (e.g., end of the month management reports, monthly reports, quarterly reports, and year-end reports). |
| Compliance Testing | A type of testing that verifies that a collection of software and hardware fulfills given specifications. For example, these tests will minimally include: "core specifications for rehosting - ver.1.5-draft 3.doc", Section 508 of The Rehabilitation Act Amendments of 1998, Race and Ethnicity Test, and VA Directive 6102 Compliance. It does not exclude any other tests that may also come up. |
| Component Integration Testing | Testing performed to expose defects in the interfaces and interaction between integrated components as well as verifying installation instructions. |
| Configuration Testing | A type of testing concerned with checking the programs compatibility with as many possible configurations of hardware and system software. In most production environments, the particular hardware specifications for the client workstations, network connections, and database servers vary. Client workstations may have different software loaded, for example, applications, drivers, and so on hand, at any one time; many different combinations may be active using different resources. The goal of the configuration test is finding a hardware combination that should be, but is not, compatible with the program. |
| Contention Testing | A type of performance testing that executes a test that causes the application to fail with regard to actual or simulated concurrency. Contention testing identifies failures associated with locking, deadlock, race conditions, priority inversion, data loss, loss of memory, and lack of thread safety in shared software components or data. |
| Data and Database Integrity Testing | A type of testing that verifies that data is being stored by the system in a manner where the initial storage, updating, restoration, or retrieval processing does not compromise the data. This type of testing is intended to uncover design flaws that may result in data corruption, unauthorized data access, lack of data integrity across multiple tables, and lack of adequate transaction performance. The databases, data files, and the database or data file processes should be tested as a subsystem within the application. |
| Documentation Testing | Documentation testing is a type of testing that should validate the information contained within the software documentation set for the following qualities: compliance to accepted standards and conventions, accuracy, completeness, and usability. The documentation testing should verify that all of the required information is provided in order for the appropriate user to be able to properly install, implement, operate, and maintain the software application. The current VistA documentation set can consist of any of the following manual types:  Release Notes, Installation Guide, User Manuals, Technical Manual, and Security Guide. |
| Error Analysis Testing | This type of testing verifies that the application checks for input, detects invalid data, and prevents invalid data from being entered into the application. This type of testing also includes the verification of error logs and error messages that are displayed to the user. |
| Exploratory Testing | A technique for testing computer software that requires minimal planning and tolerates limited documentation for the target-of-test in advance of test execution, relying on the skill and knowledge of the tester and feedback from test results to guide the ongoing test effort. Exploratory testing is often conducted in short sessions in which feedback gained from one session is used to dynamically plan subsequent sessions. |
| Failover Testing | A type of testing test that ensures an alternate or backup system properly "takes over" (i.e., a backup system functions when the primary system fails). Failover Testing also tests that a system continually runs when the failover occurs, and that the failover happens without any loss of data or transactions. Failover Testing should be combined with Recovery Testing. |
| Installation Testing | A type of testing that verifies that the application or system installs as intended on different hardware and software configurations, and under different conditions (e.g., a new installation, an upgrade, and a complete or custom installation). Installation testing may also measure the ease with which an application or system can be successfully installed, typically measured in terms of the average amount of person-hours required for a trained operator or hardware engineer to perform the installation. Part of this installation test is to perform an uninstall. As a result of this uninstall, the system, application and database should return to the state prior to the install. |
| Integration Testing | An incremental series of tests of combinations or sub-assemblies of selected components in an overall system. Integration testing is incremental in a successively larger and more complex combinations of components tested in sequence, proceeding from the unit level (0% integration) to eventually the full system test (100% integration). |
| Load Testing | A performance test that subjects the system to varying workloads in order to measure and evaluate the performance behaviors and abilities of the system to continue to function properly under these different workloads. Load testing determines and ensures that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (e.g., response times, transaction rates, and other time-sensitive issues). |
| Migration Testing | A type of testing that follows standard VistA and HeV-VistA operating procedures and loads the latest .jar version onto a live copy of VistA and HeV-VistA. The following are examples of the types of tests that can be performed as part of migration testing:   * Data conversion has been completed * Data tables are successfully created * Parallel test for confirmation of data integrity * Review output report, before and after migration, to confirm data integrity * Run equivalent process, before and after migration |
| Multi-Divisional Testing | A type of testing that ensures that all applications will operate in a multi-division or multi-site environment recognizing that an enterprise perspective while fully supporting local health care delivery. |
| Parallel Testing | The same internal processes are run on the existing system and the new system. The existing system is considered the “gold standard”, unless proven otherwise. The feedback (expected results, defined time limits, data extracts, etc.) from processes from the new system are compared to the existing system. Parallel testing is performed before the new system is put into a production environment. |
| Performance Monitoring Testing | Performance profiling assesses how a system is spending its time and consuming resources. This type of performance testing optimizes the performance of a system by measuring how much time and resources the system is spending in each function. These tests identify performance limitations in the code and specify which sections of the code would benefit most from optimization work. The goal of performance profiling is to optimize the feature and application performance. |
| Performance Testing | Performance Testing assesses how a system is spending its time and consuming resources. Performance testing optimizes a system by measuring how much time and resources the system is spending in each function. These tests identify performance limitations in the code and specify which sections of the code would benefit most from optimization work. Performance testing may be further refined by the use of specific types of performance tests, such as, benchmark test, load test, stress test, performance monitoring test, and contention test. |
| Privacy Testing | A type of testing that ensures that (1) veteran and employee data are adequately protected and (2) systems and applications comply with the Privacy and Security Rule provisions of the Health Insurance Portability and Accountability Act (HIPAA). |
| Product Component Testing | Product Component Testing (aka Unit Testing) is the internal technical and functional testing of a module/component of code. Product Component Testing verifies that the requirements defined in the detail design specification have been successfully applied to the module/component under test. |
| Recovery Testing | A type of testing that causes an application or system to fail in a controlled environment. Recovery processes are invoked while an application or system is monitored. Recovery testing verifies that application or system, and data recovery is achieved. Recovery Testing should be combined with Failover Testing. |
| Regression Test | A type of testing that validates existing functionality still performs as expected when new functionality is introduced into the system under test. |
| Risk Based Testing | A type of testing based on a defined list of project risks. It is designed to explore and/or uncover potential system failures by using the list of risks to select and prioritize testing. |
| Section 508 Compliance Testing | A type of test that (1) ensures that persons with disabilities have access to and are able to interact with graphical user interfaces and (2) verifies that the application or system meets the specified Section 508 Compliance standards. |
| Security Testing | A type of test that validates the security requirements and to ensure readiness for the independent testing performed by the Security Assessment Team as required by the Assessment and Authorization Process. |
| Smoke Test | A type of testing that ensures that an application or system is stable enough to enter testing in the currently active test phase. It is usually a subset of the overall set of tests, preferably automated, that touches parts of the system in at least a cursory way. |
| Stress Testing | A performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This failure typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources (e.g., database locks or network bandwidth), although some of these tests are usually addressed under functional and load testing. Stress Testing verifies the acceptability of the systems performance behavior when abnormal or extreme conditions are encountered (e.g., diminished resources or extremely high number of users). |
| System Testing | System testing is the testing of all parts of an integrated system, including interfaces to external systems. Both functional and structural types of testing are performed to verify that the system performance, operation and functionality are sound. End to end testing with all interfacing systems is the ultimate version. |
| Usability Testing | Usability testing identifies problems in the ease-of-use and ease-of-learning of a product. Usability tests may focus upon, and are not limited to: human factors, aesthetics, consistency in the user interface, online and context-sensitive help, wizards and agents, user documentation. |
| User Functionality Test | User Functionality Test (UAT) is a type of Acceptance Test that involves end-users testing the functionality of the application using test data in a controlled test environment. |
| User Interface Testing | User-interface (UI) testing exercises the user interfaces to ensure that the interfaces follow accepted standards and meet requirements. User-interface testing is often referred to as GUI testing. UI testing provides tools and services for driving the user interface of an application from a test. |