AOC Test Management Tool - Requirements	**< Enter Vendor Name(s) Here >** In the **Response** column, indicate how your product complies with the AOC requirements using the following response key. **4**=Out of Box (in Comments/Notes column indicate version). **3**=Future release (in Comments/Notes column, indicate version and planned release date). **2**=Gap: Requires customization (in Comments/Notes column, indicate estimated cost of customization). **1**=Gap: Other (in Comments/Notes column, explain your proposed tool). **0**=Not available in this tool. Tip: Move cursor over red triangle in the upper right corner of any cell in the Response column to view Response Key.

From any AOC location, the Test Management Tool must:				
Category	**No.**	**Requirement**	**Response**	**Comments/Notes**
I. User Interface				
	1	Provide an encrypted web-based user interface as well as desktop client interface.		
	2	Supports multiple browsers, including IE 6, IE 7, and Firefox 2.		
	3	Provide intuitive user interface so the average user can use the system without having to get specific training.		
	4	Provide short cuts using image icons and/or navigational links.		
	5	Adhere to Section 508 standards for disabilities. Standards shall apply to both test repository output and to the solution interface used by administrators and content developers (1194.21 and 1194.22).		
	6	Spell check text.		
	7	Allow for intelligent case enumeration so that a number assigned means something to the user. For example, ISB12302232007. This number provides the interface mnemonic, case number and date created (an 'intelligent' number vs. non-intelligent number).		

	8	Provide the ability to print the entire screen. Users should not have to scroll right or up/down to get a full screen print.		
	9	Syntax check scripts and flag errors.		
	10	Auto updates of newer version and support.		
	11	Allow user administration from the user interface. Administrators are allowed to assign user id's to various resources, by project. For example, an Administrator can assign a resource to use the tool for X project and can also assign another resource access to test for Y project.		
II. Test Planning				
	1	Support multiple versions of the target applications, test plans, test scenarios and test cases.		
	2	Allow for parallel testing (multiple implementations of the same application same version).		
	3	Assign time and resources to each test case.		
	4	Provide roll-up and drill down capability for all components (e.g. ability to show affected test scripts due to functionality or test script modifications; need tool to present a summary of what needs to be re-run to test the modifications).		
	5	Allow for an object-oriented approach to manage scripts in sub-components that can be combined and reused in multiple test scenarios and test types.		
	6	Provide workflow management for the testing cycles (e.g. be able to setup groups with workflow to notify for bug fix, issue clarification, retest, etc).		
	7	Support data export for analysis and manipulation in standard desktop analysis tools, such as MS Excel.		
	8	Example Documentation for creating Test Plans.		
	9	Allow automated upload of test scripts written in Microsoft applications (e.g. Excel, Word).		
	10	Provide functionality for user-friendly requirements traceability (i.e. from requirements to test script).		

	11	Allow automated upload of requirements written in Microsoft applications (e.g. Excel, Word).		
III. Test Requirement Management				
	1	Store the expected functional result of a test case and compare it with the actual result.		
	2	Trace test scripts to requirements. That is, users should be able to navigate from test scripts to requirements, provided scripts and requirements are linked during creation.		
	3	Test script preparation, setup, and execution do not require coding skills.		
	4	Scripts version control and comparison.		
IV. Test Execution				
	1	Provide ability to “time” various test cases and determine whether the result arrived within a predetermined tolerance (SLA).		
	2	Log the results of each test performed.		
	3	Log the amount of time required to perform a given test.		
	4	Provide multiple ways to create test scripts, including recording and playback, script coding, instruction for manual script execution.		
	5	Group reusable test cases in various ways to facilitate automatic replay of scenarios (e.g. the tool can replay a group of tests to form the Regression, Cut-Over and Turn-Up tests).		
	6	Progressively load the application by allowing “virtual users” to run the test cases.		
	7	Provide a means to progressively increase load and stress on individual applications while simultaneously testing multiple applications. That is, the tool needs to provide the ability to test multiple, user-loaded applications to ensure they will work together at their individual peak loads. This means the tool needs to provide for progressively increasing the number of virtual users per application as they launch test cases & execute tests, and all scriptable.		

8	Support the process of verifying and logging test results while not interfering with performance.		
9	Handle enough virtual users to simulate real-life usage of the application and record the performance of each test case for analysis.		
10	Track progress through testing so that the project manager can review the results of the tests and track any fixes through to acceptance.		
11	Provide parameter/data-driven testing.		
12	Provide distributed load testing master/slave concept as defined in test category definitions 6, 7, and 8 (see Test Category Definitions worksheet in this workbook).		
13	Enable complete stress testing capabilities, providing visibility into application performance bottlenecks.		
14	Demonstrate support for all major products and protocols for automated testing (both regression and load testing), including but not limited to: mySAP, SAPGui, HTTP, HTTPS, XML, SOAP (Web Services), UDDI, Citrix, JavaApplet, Java Web Start, Oracle SQL*NET, Oracle Finance, PeopleSoft, J2EE, .NET, JMS, SMTP, POP3 IMAP, FTP, SFTP, LDAP, ICMP.		
15	Supports testing of web-based applications, including HTML-based applications, Java applets, and Flash-based applications.		
16	Supports testing of Windows client applications.		
17	Provide robust end-user messaging (information, warning and error messages) in a clear and understandable language.		
18	Provide ability to schedule and run tests on multiple remote machines.		
19	Auto emailing option with test execution start and end time with date stamps.		

	20	Allow for email notifications when defects are opened, closed, updated, edited. Include basic information that can be dynamically selected, such as who edited, opened, etc; what number did they edit, open etc; when did the edit, etc take place; comments.		
	21	Provide robust visual graphics and statistics that monitor the servers and application performance when performance, load, or stress testing. Provide ability to export the data and print out the statistics.		
	22	Provide support for SAP Solution Manager and CATT (Computer Aided Testing Tool).		
	23	Provide the ability for a user to create test data using scripts (e.g. SQL). Users should be able to store and rerun the same scripts in the future to create new subsets of test data that will be used in test cases and performance testing. E.g. Create a SQL script that would generate test data. Store the script so that it can be rerun at a later date to generate more test data for that application, or can be modified for use in other application tests.		
V. Reporting				
	1	Provide out-of-the-box reporting.		
	2	Provide access to view data and statistics across multiple projects, vendors, implementation sites, applications etc., i.e. "slice-n-dice" data as needed (ad hoc reporting).		
	3	Include a robust reporting capability, with the majority of reporting needs covered with canned reports and the ability to create and publish custom reports to cover AOC specific needs.		
	4	Describe the level of expertise required to create custom or ad hoc reports.		
	5	The System shall provide a query building tool that supports formulation of complex queries.		
	6	Schedule and distribute reports automatically via email.		
	7	Provide a link to access and view reports online.		

	8	Create reports based on any point in time (current or past or range of dates).		
	9	Create an audit trail.		
	10	Audit trail reports must include UserIDs.		
	11	Export reports into MS-Office, PDF and HTML formats.		
	12	Create "WHAT - IF" scenarios.		
	13	Report on history within a single test cycle and across multiple test cycles.		
	14	Provide dashboard reporting.		
	15	Defect reports can be compared – can show differences.		
	16	Web-based archive of test result reports.		
VI. System Requirement				
	1	Provide comprehensive online help.		
	2	Adhere to section 508 standards for disabilities. Standards shall apply to both test repository output and to the solution interface used by administrators and content developers (1194.21 and 1194.22).		
	3	Run on Sun Solaris, Linux, or Microsoft Windows operating systems.		
	4	Easily scale to accommodate hundreds of concurrent users.		
	5	Import/convert existing TestDirector™ test cases, possibly test scripts, etc. into the test repository.		
	6	Configure to support master/master - master/slave repositories where projects can create their own remote repository that periodically can be synchronized to the main enterprise repository.		
	7	Support copying/moving of development/configuration changes from one environment to another.		
	8	Function in a cluster configuration (Active/Active and/or Active/Passive).		
	9	Provide license(s) that allow for distributed use of all components for AOC employees, contractors, associated vendors and partners that are physically located in different locations.		

	10	Must support 3-tiered infrastructure (presentation, application/middleware, database).		
	11	Proven 99% uptime. Guarantee proposed solution will perform with a variety of technical platforms, plug ins, and should commit to java, html codes, etc., that may be used with the test tool, including various Java and Crystal Reporting versions.		
	12	Seamless upgrades. Test scripts should require no rework or modifications after upgrading the testing software.		
	13	Allow all testing stake holders to see current testing results status		
VII. Enterprise Infrastructure and Security				
	1	Support all components to be maintained and stored on a central repository.		
	2	Support role-based access control.		
	3	Integrate with LDAP, such as MS-Active Directory and/or Computer Associate eTrust (Netegrity/SiteMinder), for single sign-on.		
	4	Be JSR-168, JSR-170, JSR-286 compliant.		
	5	Provide mechanisms (e.g. compression) to perform efficiently over the WAN and LAN.		
	6	Be compatible with all firewall and proxy systems.		
	7	Provide ability to be monitored by standard application monitoring software, e.g. SNMP.		
	8	Provide test console encryption for all communications not related to the test execution.		
	9	Able to run somewhat quickly on a 1200 Mhz laptop computer. Launch within 10 seconds. All pages/forms should load within 5 seconds.		
VIII. Integration				
	1	Integrate workflow notification via email (e.g. SMTP).		
	2	Provide ability to be tightly coupled and integrated with other testing tools.		

	3	Automatically generate an issue ticket in the issue tracking tool when a problem in a test case has been identified.		
	4	Provide ability to integrate to other test tools, i.e. issue tracking and code control systems to provide holistic view of dependencies.		
	5	Proven to work with TRIRIGA workplace management system.		
	6	Ability to copy and paste from Microsoft Office applications and have characters preserved. For example the dashes are converted to dashes (not funny looking characters).		
IX. Vendor Technical Support				
	1	Provide a toll free number for AOC production support.		
	2	Provide staffed phone support (as opposed to an automated system) to deliver production support.		
	3	Provide tiered product support levels to choose from during normal business hours.		
	4	Provide off-hour support with advanced AOC management notice (including evenings and weekends, in person or via phone, as needed for special activities, such as conversion, critical test times for projects).		
	5	Provide an advance copy of a fixed Release Calendar to AOC Management.		
	6	Provide written confirmation (paper or electronic) for all planned releases or emergency patches (two (2) weeks in advance for planned releases and three (3) days for emergency patches). Release Notes must include known defects.		
	7	Provide software upgrade guidelines (for example, what constitutes a major release, minor release or interim release) of the product versions.		
	8	Provide software licenses that may be moved easily from one physical location to another.		
	9	Provide online tech support (chat style).		
	10	Documentation – user operational manual and training.		

Item #	AOC Test Category	AOC Definitions & Typical Order of Execution	Typical Resource Skill set Required	Application Environment
1	Unit Testing	These tests ensure that individual components (or "units") function properly. Most often this is performed in some sort of test framework in the Development environment – which imitates the other components with which this component must interact.	Developer	Development
2	Integration Testing	As the components are integrated together, these tests ensure that they work together as expected. This is often a multi-staged integration; in which trenches of the components are rolled together to make larger components … and so on, until the whole application is integrated. The early stages of integration might happen within the Development environment. The major part of this testing is performed in the Testing environment; in which the larger "conglomerate" components are integrated.	QA Analyst / Developer	Development/Testing
3	Functional Testing	A level of testing that starts in the Development environment as unit testing then moves to the Testing environment. Testing ensures the required functionality of applications passes test plans with use cases based on business requirements. Use cases are tested, including its variants and exceptions.	QA Analyst	Development/Testing
4	Enterprise Integration Testing	Enterprise testing ensures that the underlying hardware, databases, and other third-party software or services are installed correctly and function properly together. This test phase will ensure the next stages of test can be executed and all the monitoring tools and scripts run properly.	QA Analyst / Developer	Testing
5	Regression Testing	This is a (sub-) set of Functional tests and Stress tests that can be repeated to verify that the application is still working after some change has taken place (e.g. code changes, firmware patches, etc.). Ideally, these tests are executed with automated testing tools for both Functional and Stress testing. The ease of re-application of these tests means that they can be used in whichever pre-live environment and whenever they are required. Usually the Functional tests are developed for the Testing environment and then they are captured in automated scripts to form Regression tests that are repeated in Staging or in Testing as required. Both Functional and Stress tests are combined in these tests.	QA Analyst	Testing/Staging
6	Stress Testing	As the name implies, tests are applied to stress an application, its environment and data systems, etc., under simulated user volumes that impact performance and recovery. The idea is to progressively put more stress on an application (increasing number of users, transactions, size of the data requests, etc.), until they break, no longer perform or are stressed to capacity. The tests are used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. Some testing may occur in the Testing environment, yet majority happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Testing/Staging

7	Load Testing	Testing of applications and infrastructure to measure the effects of a system's current and expected user capacity based on the current configuration, the AOC's performance criteria, Service Level Agreement or Vendor Contract, and projected usage. The results determine a 'load baseline' for current user capacity and projects what will be required to accommodate increased user capacity over the course of time. The majority of testing happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Staging/Production
8	Performance Testing	Tests determine if the system will meet the performance terms and conditions stated in the AOC's Service Level Agreement or Vendor Contract. Tests verify that the expected maximum use of the system falls within tolerance, that the system fails in a reasonably graceful manner, and that performance is acceptable. Also, performance testing is used to test response times for schema validation, overall latency/response time, availability. Majority happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Staging/Production
9	Fail-over Testing	To meet a 99.9% up-time, many hardware components are doubled up. These redundant components are useless however, unless they can take over in a predictable and acceptable way. These tests show how the redundancy works and help identify any additional steps necessary to recover. These tests should be included in the Regression test suite as appropriate (e.g. if redundant components are in Staging). Usually, this test requires that an active server is unplugged (or otherwise assaulted) to verify that its (hitherto) passive counterpart takes over as predicted. In a load-sharing scenario (active:active), the test is to verify that the remaining server takes over the whole load. This test should happen in Staging (or in Production if the Staging environment is scaled-down).	Developer	Staging/Production
10	Disaster Recovery	This test is typically performed periodically to ensure that business/operations can continue in the event that the primary data center/systems implementation is down due to a major disaster, e.g., earthquake, flood, fire, etc. The terms laid out in the disaster recovery (DR) specification are tested; this environment will not typically provide the same performance as the original environment. Tests used for the DR environment will need to be downsized to test the DR environment.	Developer, Business Subject-Matter-Expert, QA Analyst	N/A
11	User Acceptance Testing (UAT)	Acceptance criteria are typically defined in the design phase. As the previous test categories complete in order to 'accept' the progression of development and testing, this level of testing occurs at the end of the development phase, typically as a review gate or milestone test to pass prior to product launch. Various sub-sets of the above tests may be agreed to up-front in order to accept that various stages of the development are complete.	Business Subject-Matter-Expert, QA Analyst	Staging/Production
12	Cut-Over Testing	This testing is usually a subset of Regression testing. It provides for a sub-set of Regression tests in a new Production environment before giving the application approval to 'go live.' Care is taken not to change any 'live' production data, or to ensure any changes to live data are reversed. Cut-over testing occurs in Staging and also gives assures the network port from Staging to Production happened correctly.	QA Analyst; Business Subject-Matter-Expert	Production

13	End-to-End Testing	Testing that involves test monitors that perform end-to-end tests of each application of a system (typically from the CCTC to a court). This level of testing is to determine if and when any function or system integration point breaks prior to system implementation. Tests for the application and network through the Court's network to the CCTC network are executed. Functional and non-functional testing are included ensuring security requirements are met. Output facilitates ongoing monitoring to ensure all production systems are working as expected.	Developer	Staging/Production
14	Product Acceptance Testing (PAT)	This is the set of tests from all tests above that determines whether the product is viable based on the agreed requirements. While delivering a product to meet both functional and technical requirements is the goal, these requirements can only be validated through PAT. Acceptance criteria, therefore, tends to tag on more flexible statements as well as referencing rigorous testing. For an application provided by a third party, these tests allow the AOC to decide to go ahead or stop a new application. Typically a 0:0:10 criteria is used, which allows 0 grade 1 errors, 0 grade 2 error and up to 10 grade 3 errors. For an internal development, these tests determine whether we are ready to move onto the next stage of implementation.	Business Subject-Matter-Expert, QA Analyst	Staging/Production
15	Localization	This tests the various concurrent builds of an application individually. Some applications are customized (localized) for different groups of users. Common functionality can be tested in common – per the above tests – but localized functionality should be tested in each build as appropriate. Several versions of Regression tests are required: to test that a localized version maintains its core functionality and to test the "delta" of each localized version.	Developer, QA Analyst	Testing, Staging/Production
16	Turn-up Testing	When a court is first given access to an application housed at the CCTC, these tests are used to determine whether the court has suitable access to the application. Various tests can be used, from: • Testing the bandwidth available to/from the CCTC • Running the Regression Tests from the court to determine whether all functionality works • Running a Load Test from the court (which may be covered in a good Regression Test) to determine the performance.	Technical SME	Staging/Production
17	Data Conversion	When a new court is on-boarded, data from previous systems is often ETL'd into the new application environment. Data conversion testing determines whether that data has been uploaded correctly. Typically, this involves a manual process in which information is accessed via both the previous and the new applications to see whether it matches. Since the new environment may require different data architectures, direct database comparison is not always viable and the two versions need to be compared by accessing data in either version via its user interface.	Developer, QA Analyst, Business Subject-Matter-Expert	Staging/Production
18	Usability	A representative group of business users use the system to see how it functions relative to their expectations. Typically the testing occurs after training as the users try out features of the application intuitively and have the opportunity to validate that their requirements have been interpreted correctly. Usability testing gives the Developers valuable insight from the end users prior to UAT and occurs in the Testing environment.	Business Subject-Matter-Expert, QA Analyst	Testing

19	System	This is the final stage of (non-enterprise) integration testing, in which all the components have been integrated and the whole application (except links to other enterprise functionality) is available.	QA Analyst	Testing
20	Test Data	Testing an application invariably changes the environment in which the test occurs. Steps can be taken to script “reset” tests that return the environment to the original state. E.g., if a test changes a user’s password, the equivalent reset test must change it back (or cycle through the necessary iterations before it can return the original). Also, another approach is to install a fresh set of test data before each test commences. In this way, all the parameters used in the test cases can be re-used, since they relate to this clean version of the data. This process involves: 1. Backing up any live data 2. Loading the Test Data 3. Running the test 4. Analyzing any end-state data 5. Re-loading the original live data. One of the reasons for creating cut-over tests, rather than running a full Regression test, is to remove any tests that change data that can’t be reset by running another test (e.g. it may not be possible to un-approve a workflow that was approved in a test). This allows the cut-over test to be run in a live environment without compromising the data and without havin	Developer, QA Analyst	Staging/Production