UAT Test Plan

[Subject]

Date

Client Name [from the engagement letter]

Title

Company Name

Address

City, State Zip Code

Date

Dear Name

[Insert text here.]

Very truly yours,

PricewaterhouseCoopers LLP

Guidance:
High Quality Deliverables

- Are impactful, direct, compelling, persuasive & specific to the client's business.
- Are well organized and free from grammatical, mathematical and typographical errors.
- Are clear and concise.
- Are fact based with substantiated assumptions and as appropriate, reference supporting documentation.
- Demonstrate clear reasoning (deductive or inductive).
- Are sufficiently reviewed by partners, directors, managers, SMS, etc. as appropriate.
- Leverage prior distinctive work within the practice and firm.
- Leverage Global Best Practices and Sector Business Models where appropriate.

Guidance:
Information contained within the deliverable should be:

- Reliable – information and processes used should be gathered, recorded, analyzed for reliability and disclosed in a way that conveys the quality and materiality of the information.
- Clear – present information that is understandable and usable by stakeholders.
- Balanced – content should represent an unbiased reporting of performance, if appropriate; include both favorable and unfavorable results.
- Accurate – Information presented should be fact based and supported by relevant documentation.
- Timely – Information should be current.
- Thorough – data collected be should sufficient (e.g. time period covered, geography, accounts, processes etc.).

PricewaterhouseCoopers LLP, Street, Town, County PostCode, Country
T: +44 (0) 123 4567, F: +44 (0) 123 4568, www.< >.co.uk

Authors

This document was prepared by:

John Doe, Release Test Manager CIO Group Address Toronto, ON T: 416.364.5000 F: 416.364.5000 john.doe@abc.ca	**John Doe, UAT Test Manager** CIO Group Address Toronto, ON T: 416.364.5000 F: 416.364.5000 john.doe@abc.ca	**John Doe, SIT Test Manager** CIO Group Address Toronto, ON T: 416.364.5000 F: 416.364.5000 john.doe@abc.ca

REVISION HISTORY

The softcopy of the most current version of this Test Plan can be found *file location*.

Version Number	*Revision Date* *mm/dd/yyyy*	*Summary of Changes*	*Document Author*

APPROVAL LIST

The following stakeholders agree that the information reported in this Test Plan is accepted and approved. The information in this document may be used in test preparation and test execution.

Version Number	*Revision Date* *mm/dd/yyyy*	*Department*	*Name and Role*	*Approval*	
				E-mail	*Physical Signature*

DISTRIBUTION

The softcopy of the most current version of this Test Plan has been distributed to the following individuals.

Department	*Name*	*Title*

DOCUMENTATION

The following documentation was utilized to create this Test Plan.

Document Name	*Document Location*

Table of Contents

NOTE: The table of contents above is matched to the section outline below. No additions or modifications should be required on this page. To update the table of contents to reflect current page numbers and modified section titles, place the mouse pointer in the table of contents and right click. Select Update Field, select Update entire table, and click OK.

1. Introduction

***NOTE IMPORTANT**: This is a User Acceptance Test Plan template to support the various test types of User Acceptance Tests that a release may require. The Testing Strategy deliverable provides an overview of the test types to be included in User Acceptance Test and should be referenced, rather than the information repeated here. In general, any information that is documented in the Testing Strategy document that will be unchanged for User Acceptance Testing should simply be referenced in the User Acceptance Test Plan. For small projects, there may not be an overall Testing Strategy document. In this case, it is important to note any information that will be true for other test stages, such as the approach, the test procedures, severity definitions, etc.*

1.1. Purpose

Define the purpose of this document that is to be the User Acceptance Test Plan for a given release. This document should provide the details of how User Acceptance Testing will be organized and executed.

This…

1.2. Intended Audience

Describes the audience of this document, who will use the document and how they are intended to use it. In general, the User Acceptance Test Plan is intended for use by the test team to provide guidance when preparing the test cases, scripts, data, test procedures, and to ensure complete coverage of the requirements. Other audience members may include project management and other project stakeholders.

The intended audience…

1.3. Approach

Describe how this plan was determined. Who, what, where, when and how.

Describe the overall approach to User Acceptance Testing. What testing will be done first? Last? How will the data be managed in the Databases? If there are Packaged Applications involved, how will they be managed? Are there any dependencies or resources required from other test stages? What are the risks to successfully complete the tests and how are they mitigated?

Why is this the right strategy that will produce accurate results?

What are the reasons this approach was taken as opposed to others that were considered? What may be other approaches that should be pursued in the future or built upon this testing?

NOTE: This may already be documented in the Testing Strategy document. Therefore, this section may be a brief summary or may simply reference the Testing Strategy document.

This…

2. Testing Overview

Any principles upon which the test plan is based should be described here; this can include how test cases are traceable to requirements, how the test effort attempts to emulate the production conditions etc.

The Testing Overview is…

2.1. Testing Quality Assurance Approach

This section should explain the approach to quality assurance during this test stage. How will quality be monitored? Will a readiness review be performed? Who will review variances and issues? How often? Additionally, the environment should be discussed. For example, there are typically some differences in the test environment and production. There should be a description of the steps taken to limit those differences (e.g. use data from production), where there are gaps and how the potential exposure will be managed.

The following documents should be included:

- Peer Review Schedule
- Deliverable Review Matrix
- Quality Gate Close Out Memos
- Lessons Learned and Best Practice Sessions

This…

2.2. Assumptions

List any assumptions specific to the testing of the solution under test for the current project. Examples include, testing team size, resource availability, skill level, team roles & responsibilities, service level agreements internal and external to the team, test readiness of the application, etc.

This…

2.3. Organizational Chart

This section may include an organization chart of who is performing the testing, reviewing the results, ensuring quality, and who certifies test acceptance and readiness for movement to UAT.

This…

2.4. In Scope

List here all the requirements and/or functional areas that will be included in User Acceptance test execution. These will primarily correspond to requirements documentation. For solutions with external interfaces, list all external interfaces that will be tested.

Guidance:

- If applicable, indicate why we are not doing a full report e.g. report is part of a bigger report etc.
- Consider whether a detailed scope should be provided here or in the Appendix.
- Consider indicating any changes to the scope, such as procedures we were not able to complete, in either the Executive Summary or the body of the report.
- When scope was limited provide explanation of limitations and provide necessary qualifications.
- Document any assumptions.

Function/Requirement	*Requirements Reference(s)*	*Notes*	*Mandatory to Pass*
Feature id/area	Requirements document and/or section	Any exclusions	Document if this Function / Requirement is mandatory to pass or not

In Scope Test Areas

2.5. Out of Scope

List all requirements or functional areas that will be not be tested as part of the test stage. For example, it should be noted where a requirement or functional area will be exercised as part of a different test stage, different testing group, or a subsequent release.

Function/Requirement	*Requirements Reference(s)*	*Notes*
Feature id/area	Requirements document and/or section	Any exclusions

Out of Scope Test Areas

2.6. Test Case Specifications

Describes how the test cases will be developed and organized. If possible, indicate how many test cases will be written and their relationship to the requirements and design i.e. at one to one relationship with the rows in the 'in scope' table.

The test case specifications…

2.7. Testing Schedule

This should provide the initial estimate of the duration and schedule of User Acceptance Testing and the relationship to the other test stages. The information should compliment that which has been included in any project plans and reference such plans. Rather than copy the entire schedule, this section should reference where the schedule is maintained.

Include information on any shifts or any other type of time sharing arrangement perhaps driven by the need to conduct tests that require exclusive use, or other resources (legacy systems etc.) that are only available for certain days/periods of a day.

This…

2.8. Test Automation

This section should state whether test automation will be utilized. If it is not, the reasoning should be described. If automation will be utilized, a general description of any automation should be given here, why it is being done, the general timeline, test case creation information, how test data will be generated, etc. Is it being used for a particular type of testing (regression being the most common use)?

Test Automation is …

2.9. Test Case Identification

This section should describe how test cases will be identified and how each will be classified. Document how Test Case identification would be done differently, if different from the ETM Policies and Procedures document, section 6.

Test cases should be identified by…

2.10. Test Case Pass/Fail Criteria

Specify the criteria to be used to determine whether each test case has passed or failed. For example, to pass a test, a test item must correctly exhibit all features of the test case, as verified through the test scripts. Additionally, all test scripts related to the test case must be executed and passed for the test case to pass.

Pass/Fail Criteria…

2.11. Variance Classification

Describe here how variances will be classified. Minimally there is a severity classification; often there is an additional classification to indicate the priority of the variance correction and the cause or area of the variance (i.e. incorrect documentation, missed requirement, etc.). If these are already identified in the Testing Strategy then reference the appropriate section of that document.

Sample text is provided. Edit, add and revise as necessary.

Variance Management Severity Level Definitions	
1 – Showstopper	• Item under test fails or gives incorrect results. Testing cannot continue until the variance is repaired • Unable to proceed with complete application / component testing • Major impact to test / project schedule • Unacceptable impact to business or technical processing; work around is not available • Analysis within 2 hours • Target resolution within 24 hours or negotiation required
2 – Critical	• Item under test fails or gives incorrect results. Testing can continue by bypassing the item that caused the variance. The variance must be repaired for the current release or the defective function must be removed for the current release • Unable to complete test cases in the application / component

	• Minor impact to test / project schedule • Unacceptable impact to business or technical processing; by-pass or temporary work around is available • Analysis within 4 hours • Target resolution within 48 hours or negotiation required
3 – Non-critical	• Item under test does not fail, but gives an incorrect result, which does not affect a business or operational function. Testing can continue. Repair of the variance can be deferred to a subsequent release • Cosmetic change • Item is not critical • Minimal impact to the application / component • Minimal impact to the test schedule • By-pass or permanent work around is available • Analysis within 1 day
4 – Live With	• A workaround solution exists and has been tested. The code may be implemented. The variance should be fixed as soon as possible • Unexpected application behavior which does not affect critical functionality • Incorrect data in a non-critical report • Incorrect functioning in a non-critical function
5 – Other	• New requirements - testing identified a new Business Requirement • Testing identified a missed requirement

Also describe here how severity and priority will be set and the process by which the severity can be changed.

The required variance information should be described here or the policy and procedures document should be referenced:

- Variance ID
- Short description
- Long description
- Severity (1,2,3,4)
- Priority (H, M, L)
- Opened by (name)
- Stage found in
- Area found in
- Assigned to
- Target completion date

- Comments (may be used to track ongoing comments)
- Resolution

2.12. Training Needs

Identify training needs and options for ensuring the test team has the necessary skills Reference the project plan if it has been documented there.

This…

2.13. Test Tools

Provide a list of any tools that will be used, their purpose, and if applicable when they will be used. If the tool impacts the testing strategy or process indicate how so. Test tools can include: test automation tools, data management, load simulators, metrics tools and reports, test case management (track to requirements, track pass/fail etc.)

The test tools being used are….

2.14. Risks and Mitigation Strategy

Identify potential risks and how they will be mitigated during testing.

This…

3. Test Environment

The test environment must be carefully documented so that results are reproducible and others can use the data as a reference taking into consideration any variables. If there is a document that describes the standard operating environment on the project reference that document here rather than repeat all the detailed information (components, management, etc.). Only specify here what isn't documented elsewhere and reference the other related documents. The Environment Manager should maintain a single document that defines the correct software versions etc. If the Testing Strategy provides information that is different from the environment document for the project, and that is information relevant to this test plan, reference that here as well. Note: The purpose of this section is not to restate the information in the Testing Strategy or other environment documentation.

The test environment…

3.1. Overview

Overview of the test environment (include picture that shows the relationship of hardware and all software components). Carefully document anything that is different from the production system.

This…

3.2. Test Environment Management

Describe (or reference the document that describes) who will coordinate the potentially many pieces of the test environment, schedule exclusive use of limited resources, etc. Also describe how the controls of the environment will work. For example who will have logins, who will authorize and execute changes to the environment?

This…

3.3. Environments

Note the environments in which testing will occur.

3.4. Components

Describe each component in the picture.

The components include…

3.4.1. For each machine:

- Document all software release levels (include patch levels)
- Document any operating system tunable parameters
- Document memory, CPU specifications (model, MHz, number of, etc.)
- Document number disks, partitioning, swap space.
- Are any file systems mounted remotely? Any redundancy? Number of controllers? Any other notable conditions, etc.

Machine components…

3.4.2. For each database:

- Document which applications use the database (i.e. a dedicated database or shared)
- Document all database parameters, raw I/O or use of a file system, any cache's temp space allocation, all tunable parameters (number of server processes, etc.), etc.
- Describe any backup and journaling, clustering, connections to business continuity centers (warm stand-by, etc.)

Database components…

3.4.3. For any network:

- Document its configuration (IP addresses, routers bridges, etc.)
- Document bandwidth.
- Document if it is shared in any way, if it is shared must minimize impact (document how).

Network components…

3.4.4. For each piece of middleware in the solution (integration brokers, etc):

Document all tunable parameters or any other part of it that is configurable.

This…

3.4.5. For each component of customized code:

Document the version of the code and the values of any configurable parameters, etc.

This…

3.4.6. For each Packaged Application in the solution:

- Describe the characteristics of the interface.
- Is it single-threaded or multi-threaded?
- Is the communication synchronous or asynchronous?
- If it is asynchronous, can there be several outstanding requests simultaneously?

This…

3.4.6.1. If the real application is being used document:

- Release level
- How data will be managed in the database (is it representative of the production environment
- Why the data used will not skew results (i.e. using very similar data may result in the disk caching, etc. by the operating system that will not happen in production)
- Data link (speed of link, protocol)
- Any protocol conversion being done and how it is being done.

This…

3.4.6.2. If the application is being stubbed out:

- Document how it is being stubbed out
- How this simulation will influence the results
- How the stub simulates application (response times, data content, etc.)

This…

3.4.6.3. Key Environment Activities:

For this section, document

- Reference number
- Task/activity
- Person/group responsible
- Timeframe (planned and actual)
- Status (complete, in progress etc.)

If the table already exists in the SIT Test Plan then simply refer to the appropriate section within the SIT Test Plan here.

The table below provides examples of the types of tasks that are to be documented.

Ref #	*Task/Activities*	*Responsible*	*Target Date*	*Status/ Comments*
U1.1	Test Director Set Up	Test Director Support – Name	**Planned:** Date **Actual:** Date	Complete
U1.2	Code Promotion to SIT environment *	Name	**Planned:** Date **Actual:** Date	Complete
U1.3	Environmental Health Check	Name	**Planned:** Date **Actual:** Date	Complete
U1.4	Environmental Shake-Down (done prior to SIT)	SIT QA Team	**Planned:** Date **Actual:** Date	Complete
U1.5	Test Data Preparation (Seeding) Automated seeding may be required. Team members to update the data file according to their needs	QA Team	**Planned:** Date **Actual:** Date	
U1.6	Restore workflow wait times for UAT	Name	**Planned:** Date **Actual:** Date	

4. Test Data

This section is made up of two subsections: Data Requirements and Data Availability. The information in these sections may be reported in the sections, recorded in the Appendix and referenced in the sections or recorded in another document and referenced in the sections.

4.1. Data Requirements

This section identifies the type, class, breadth, depth, volume and data source of test data required to effectively test the planned change in a test level. It reports the test data acquisition method i.e. how the data will be acquired, if the data requires scrubbing, etc. It describes the test data strategy and approach for test data seeding and manipulation. It also identifies any materials such as forms that need to be ordered or updated for the test. This section records if and how test data is shared amongst the different levels of testing. This section identifies the test data requirements to perform the requisite tests. The following table may be modified for use.

Test Data Type	*Test Data Class*	*Test Data Breadth*	*Test Data Volume*	*Data Source*	*Data Source Type*	*Comments*

4.2. Data Availability

This section reports the availability of the test data that is used in UAT.

5. Test Execution Guidelines

This section is intended to document where test procedures vary from the test procedures specified in the Testing Strategy and/or the ETM Roles, Policies and Procedures document.

The test execution guidelines…

5.1. Entrance / Exit Criteria

Describe the entrance criteria for this test stage. Entrance criteria may include: the SIT verification tests have been run successfully and all SIT stage checklist items have been successfully completed.

Describe the exit criteria for this test stage, including metrics as to % test cases run/passed, number of open (not fixed yet) variances of each severity etc. Exit criteria may also include: all planned tests must be run and there are no open Critical issues. Additional agreed upon criteria may be included in the exit criteria.

Test Execution Details
Entry Criteria: • The UAT environment shakedown was successfully completed and there will be no interruptions to the test as a result of the environment • Test accounts have been set up and allocated to team members • Successful completion of all required Unit, DIT, and SIT and testing by the respective teams which includes a set of test cases (agreed upon in advance by the Release Test Manager and the appropriate Test Managers) having passed with no Severity 1 or 2 variances. • Approval to Migrate (Go / No Go) into UAT • All Change Controls are either addressed or deferred • UAT Test Plan Review with sign off from the required parties • Test data has been created • Signoff of all test cases and data by the required parties • Frozen Software Packages / Code Complete

Exit Criteria:
The following exit criteria must be met before the system is ready for launch:
• All test cases have been successfully executed (passed or deferred based on approval from the business)
• All fixes have been successfully re-tested
• Outstanding variances:
  ○ Severity 1 (Critical) and Severity 2 (High) – None
  ○ Severity 3 (Medium) and Severity 4 (Low) - All outstanding variances have been agreed to by the Business as acceptable
• UAT Exit Report has been prepared by the UAT Test Manager and signed off by all required parties.
• Production Go/No Go meeting should be held where approval will be requested to move to

Production Implementation

- The UAT Exit Report summarizes the quality of the system code upon completion of UAT. The report will indicate if the application meets UAT Exit Criteria and will recommend that the application is suitable for pilot or launch.

5.2. Process

Typically the process becomes more formal and approval driven throughout the test cycle.

This...

5.2.1. Test Preparation

Describe how the various test types, as defined in the Testing Strategy, will be coordinated and executed. This should include regression and variance testing.

The initial User Acceptance Test cycle will typically not include extensive variance testing, except to verify any variances reported in production that are intended to be corrected in this release. Subsequent User Acceptance test cycles should include time for variance testing and this should be planned for. The amount of time required for variance testing will be based on the number and severity of the variances identified in the previous test cycles and the number of those corrected for the current test cycle. Estimates can be made based on the amount of customized or new code in the release and results from previous test stages, if applicable.

This...

5.2.2. Test Execution

Describe the step by step processes participants will use for conducting the tests; For example when a bug is detected will a variance be opened and by whom, how will testers track test execution progress etc.

This...

5.2.3. Variance Resolution

Describe the process for variance resolution. What information is communicated and entered, how is the variance initially tested by the implementer, what is the process to have the fix delivered, recorded and retested by the tester. If variance resolution for User Acceptance test does not differ from the ETM Roles, Policies, and Procedures, simply refer to that document here.

This...

5.2.4. Suspension/Resumption Procedures

There will be times during each stage of testing where testing will need to be halted so that fixes can be installed (incremental release) or because a severe problem is blocking test progress. In each case the process should be described regarding how this will be managed. This includes escalation procedures and how an urgent fix will be installed in a controlled way. The expected turnaround time for each severity should be described as well.

The suspension/resumption procedures...

5.2.5. Test Metrics and Reporting

This section describes the metrics that will be regularly collected and reported upon during this test phase and should include:

- What are the measurements that will be taken?
- What methods will be used? (i.e. how will they be taken?) Will the recording of measurements skew results – will code have changes to record measurements.
- What will be considered a success? What will be considered a failure?
- What other data will be gathered? (i.e. check for memory leaks, does GUI response slow while load on the servers increase, what queues will be observed looking for delays, are there platform tools, admin commands to look for bottlenecks in the flows, etc.)
- Test metrics may include: pass vs. fail, percentage of executed test cases, percentage of severity 1 and severity 2 variances raised (as part of all raised variances), feature/area test case statistics, test stage metrics, graphs plotting test case execution, passed first time, open vs. closed variances over time, pass/fail, open variances by severity over time, etc.

Test Metrics and Reporting…

5.2.6. Test Closure

This section describes the test closure process.

Test Closure…

5.3. Roles and Responsibilities

This section should document the roles and responsibilities of each member of the project. If the roles and responsibilities are identical to those documented in the ETM Roles, Policies, and Procedures document, then that should be referenced here.

This…

5.3.1. Test Specification and Execution

Describe who will be responsible for the different aspects of test preparation and execution. Reference the project plan and/or Testing Strategy for information already captured there. If not, it should be included here.

Area	*Test Specification*	*Test Execution*	*Estimate*

Testing Responsibilities

5.3.2. Test Support

Describe all the resource requirements that are in addition to the practitioners' directly managing/executing test. The management of the resources identified should be among the stakeholders of this document. This information may already be specified in the Testing Strategy document, in which case it should simply be referenced here.

Area	*Resource Need*	*Purpose*	*Estimate*
XYZ legacy system support	DBA	Establish data and perform DB clean-up after tests execute.	3 hours per day for 3 weeks

Hardware support	Support Specialist	Diagnose, Replace faulty hardware	On call 5x10.
Communications support			
System software			
Test tools			
Variance tracking system			

Additional Resource Requirements

5.4. Deliverables

List here the deliverables for this testing stage.

Candidates are:

UAT Test Plan

UAT Execution Schedule

UAT Test Cases / Test Scripts

UAT Test Data and Test Data Sheet

Daily Variance Reports

UAT Test Results (Exit Report) with screenshots where required

UAT Testing Phase Sign off form

Deliverables for this stage include...

5.5. Communication Approach

Describe the communication approach in the matrix below. Add all appropriate meetings and forms of communication and indicate with an "X" who, by role, is included in the communication. This table has example data, please review and modify as necessary.

	Communication Vehicle	*Frequency*	*Responsibility*	*Release Test Mgr*	*Release Dev Lead*	*Dev Work Package Lead*	*SIT Test Mgr*	*UAT Test Mgr*
Kick-off	Meeting	Once	Project	X	X	X	X	X

Meetings:			Manager					
Team Status Meeting	Meeting	Weekly	Project Manager	X		X	X	X
Project Plan Working Session	Meeting	Weekly	Project Manager	X		X	X	X
Walkthroughs for review and approval	Meeting	Weekly	Project Manager	X	X	X	X	X
Variance Status Review	Meeting	Daily	Project Manager	X	X	X	X	X
Project Status Reports	Email	Weekly	Project Manager				X	X

6. Candidate Additional Sections

The following sections may be appropriate to include as the last sections of a document.

The following sections...

6.1. Glossary of Terms

Guidance:

Use abbreviations sparingly in report. This section may not be required

Term	Definition
[ABC]	[Vendor]
< >	IRD Department
Engagement Letter	Letter of Engagement with Vendor dated [date]
Other	Other terms and abbreviations as needed

Provide an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand the document. This information may be provided by reference to one or more appendices or by reference to other documents.

Definitions, Acronyms, and Abbreviations...

6.2. Open Issues and Future Considerations

If there are known open issues, things that are anticipated to change that will impact this document etc., list them here describing any important attributes, considerations, timeframe, anticipated options/resolution etc.

Open Issues and Future Considerations...

6.3. References and Related Documents

If there are related documents the reader should consider or may be of interest list them here

List the title, report number, revision, date, and publishing organization of all referenced documents. Identify the sources from which the documents can be obtained. This information may be provided by reference to an appendix or by reference to another document.

References and Related Documents...

6.4. Acknowledgements

Provide any acknowledgements to others in addition to the author who were instrumental in completing this document.

Acknowledgements...

7. Appendices

7.1. Analysis of [Insert Text Here]

Term	*Definition*
Business Cycle Testing	The business cycle test will simulate production environments by emulating activities performed against the application within significant time cycles, such as start of day, mid-day, end of day, end of week, course of months, etc. This tests the end-to-end processes and workflows that occur within the business cycle. This may also be called Workflow Testing and End-to-End testing.
Concurrency Testing	Concurrency testing will verify that the application supports multiple users while they attempt to add, modify and delete records at the same time. Tests include verification of on-line usage and loader functionality. This may also include other interfaces, in addition to user interfaces, such as interfaces that get updates from batch runs, etc. This is done as part of performance testing.
Data, Data Conversion and Database Integrity Testing	This testing involves 3 areas of verification. First if the system requires data conversion, the conversion should be run and verified. This may be done as part of the Installation Test, or separately, as appropriate. Second, the data should be verified for test readiness, but also to ensure that the system is properly handling the data and changes to data. Finally, data integrity, in terms of the maintenance of the data through unexpected activity, errors and concurrent access to the data must be verified to ensure the system maintains Data Integrity, as defined by the product. Additional research into the DBMS needs to be performed to identify the tools / techniques that may exist to support the testing.
Variance Testing	Variance Testing is testing which is performed after making changes to correct variances identified in the solution. Test Cases that identified the variance are rerun to ensure they pass. As well some regression testing may be required to ensure no other unit of software has been impacted by the changes.
Documentation Testing	Documentation Testing is testing that verifies the information provided in product documentation, including user guides, installation and configuration guides, troubleshooting guides, administration and other user, technical and support documentation. This testing may be done concurrently with Functional testing, where the functional test cases follow the documentation.
Environment Configuration Testing	Environment Configuration testing verifies the operation of the System under test on different software and hardware configurations. 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 (e.g. applications, drivers, etc.) and at any one time many different combinations may be active and using different resources. In general, Environment Configuration testing should be based on the client's planned hardware and software configuration and recommendations for that configuration. Broader Environment Configuration testing can become expensive.
Failover and Recovery Testing	Failover / Recovery testing ensures that the System under test can successfully failover and recover from a variety of hardware, software, or network malfunctions without undue loss of data or data integrity. Failover testing

	ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly "take over" for the failed system without loss of data or transactions. Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions (or simulated conditions) to cause a failure, such as device I/O failures or invalid database pointers/keys. Recovery processes are invoked and the application/system is monitored and / or inspected to verify proper application/system/data recovery has been achieved. Disaster Recovery Testing requires testing a full disaster scenario.
Functional Testing	Functional testing should focus on any requirements for test that can be traced directly to use cases (or business functions or business requirements), and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques, that is, verifying the application (and its internal processes) by interacting with the application via the GUI and analyzing the output (results). This includes testing boundary conditions, exception handling, error message validation, and negative testing. Functional test cases may be used to execute Business Cycle Testing, Data Integrity Testing, and other types of testing.
Installation Testing	The purpose of the Installation Test is to ensure that the software can be installed under different conditions, such as a new installation, an upgrade, and a complete or custom installation, and under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, etc.
Installation Verification Testing	This test ensures that once installed, the software operates correctly. This usually means running a number of the tests that were developed for Functional testing. If appropriate, this may include testing of data conversion. This testing may also be called Sanity Test, Smoke Test, Hurdle Test.
Load Testing	Load testing is a performance test which subjects the System under test to varying workloads to measure and evaluate the performance behaviors and ability of the System under test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (response times, transaction rates, and other time sensitive issues). Testing beyond the expected maximum is considered stress testing and is not part of the Load Test.
Development Integration Testing (DIT)	DIT will proceed once all Unit testing has been successfully completed. The primary objective of DIT is to validate that the individual programs, components, or modules of code function together as a whole correctly. During DIT, the emphasis is not on detailed functional requirements, but on the ability of the individual developers' coding to function within the DIT environment.
Non-Functional Testing	Non Functional Testing focuses on how well the application or system behaves. This includes (but is not limited to), usability testing, reliability testing, performance testing, security testing and load testing.
Operability Testing	A level of testing in which the operations of the system are validated in a test environment that is identical to, or closely simulates, the production environment. This includes verification of "production-like" JCL, installation procedures, back out and recovery procedures, and migration and conversion procedures.
Production	The primary objective of PAT is to ensure that the system changes going into

Acceptance Testing (PAT)	Production meet all non-functional requirements of the changed applications and that all necessary support requirements have been met. PAT will also ensure that the system is ready for release to Production from an operational/support perspective. PAT ensures that the system performs as expected under load, meets security requirements and that all changed applications are ready to be supported in the production environment.
Performance Profiling	Performance profiling is a performance test in which response times, transaction rates, and other time sensitive requirements are measured and evaluated. Performance profiling is implemented and executed to profile and tune a System under test's performance behaviors as a function of conditions such as workload or hardware configurations. Typically, performance profiling is done during the design phase. The output from performance profiling is the performance expectations that will be verified through Performance Testing.
Performance Testing	Performance testing is designed to validate the expected performance of a system, or determine performance limitations of a system. The goal of Performance Testing is to verify performance requirements have been achieved. It confirms that the application and technical environment will properly support the anticipated increased transaction volumes, according to an acceptable set of response or elapsed time metrics.
Regression Testing	Regression testing is testing which is performed after making a change or repair of the software. Its purpose is to determine if the change has regressed other aspects of the software. As a general principle, the changed or new software unit will be unit tested, and then be integrated into the product and test cases that address potentially impacted areas will be run.
Security and Access Control Testing	Security and Access Control Testing focus on two key areas of security: 1. Application-level security, including access to the Data or Business Functions, and 2. System-level Security, including logging into/remote access to the system. Application-level security ensures that, based upon the desired security, actors are restricted to specific functional roles/use cases or are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that user “type” one can see all customer information, including financial data, however, user two only sees the demographic data for the same client. System-level security ensures that only those actors granted access to the system are capable of accessing the applications and only through the appropriate gateways.
Stability Testing	Stability testing refers to testing the system stability by running the system for a predefined period of time and under load to ensure the continuous stable operation of the system. This may be defined as part of Stress Testing if appropriate.
Stress Testing	Stress testing is a type of performance test implemented and executed to find errors due to consumption of resources or competition for resources. Low memory or disk space may reveal variances in the System under test that aren't apparent under normal conditions. Other variances might results from competition for shared resource like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the System under test can handle.
System Integration	System Testing should be formal testing that is conducted by an independent

Testing (SIT)	test team. The primary objective of System Integration Testing is to ensure that detailed system changes have been coded according to specification without adversely impacting the existing functionality. System Integration Testing validates the functionality of the 'as-built' System. This testing is driven primarily from the design. As a result, the SIT test cases will be mapped to the Detailed Design and Use Cases to demonstrate the comprehensiveness of the testing. The purpose of System Integration Testing is to perform all positive and negative functional testing, regression testing, performance testing, volume testing, stress testing, and any necessary interface testing to validate the quality of the integrated code. All significant aspects of new functionality should be tested. SIT is the most comprehensive phase of testing. The goal of SIT is to identify and correct the most significant variances so that User Acceptance Testing can be conducted and the code released with a high degree of assurance that it will meet both the specifications in the Detailed Design and the requirements outlined in the BRD. Typically, the bulk of the variances are identified during the SIT phase.
Third Party Interface Testing	Third Party Testing verifies the integration points of the solution under test to any third party application or service. This may include other applications within the client environment, such as an accounting system, or an external source of information or user of information. In the event that the solution under test does not interface with any other application or service, this testing should be omitted.
Unit Testing	The primary objective of Unit Testing is to ensure that the code that has been developed has been thoroughly tested against design specs and existing functionality. Unit testing validates modifications to the smallest unit of the configuration or software application to verify that the functionality satisfies the design specifications. The purpose of unit testing is to verify that every executable statement can execute successfully and achieve the desired results. Unit testing is not intended to validate that end-to-end functionality is working correctly. Unit testing includes (but is not limited to) positive, negative, decision, boundary, field input, and field limit tests. The unit tests themselves are limited to the internal logic of a unit.
User Acceptance Testing (UAT)	The primary objective of User Acceptance Testing is to ensure that the system functional changes have been implemented according to business requirements without adversely impacting the existing system functionality. UAT focuses on end-to-end business flows, usability and testing specific functionality identified in the systems change. UAT validates that the end-users will be able to utilize the system to accomplish their defined objectives, as outlined in the Business Requirements Document (BRD). As a result, the UAT test cases will be mapped to the BRD to validate that the delivered system meets the defined requirements. Emphasis is on testing key user functions and trying to stress or break these key user functions. UAT will execute positive path and negative testing of system functionality as it is added to the UAT environment, including regression of previously tested system elements. UAT results in final business certification that the system changes are ready for release into the Production environment.
User Interface Testing	User Interface testing verifies a user's interaction with the software. The goal of UI Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the System under

	test. In addition, UI Testing ensures that the objects within the UI function as expected and conform to corporate or industry standards. UI Testing should include verification of the usability of the UIs based on the UI design. UI Testing may use existing Functional and Security Test cases.
Volume Testing	Volume Testing subjects the System under test to large amounts of data to determine if limits are reached that cause the software to fail. Volume testing also identifies the continuous maximum load or volume the System under test can handle for a given period. For example, if the System under test were processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.
Negative Testing	A type of destructive testing performed to confirm that the application error handling functions correctly when processing missing and / or invalid user input or data files.
Error Handling Testing	A type of testing performed to confirm that the application gracefully handles errors that occur during normal execution.

Additional Appendices

Appendices may be used to provide information published separately for convenient document maintenance (for example, classified data) or for providing additional information about, for example, testing practices and procedures. Each appendix should be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling. Appendixes are lettered alphabetically (A, B, etc.). Additional Appendices may be added as necessary.

8. Exhibits

8.1. Analysis of [Insert Text Here]

Insert text, table, or client documentation here