<Company Name>

<Project Name>
Test Plan

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process ™. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Test Plan

1. Introduction

1.1 Purpose

This Test Plan document for the <Project Name> supports the following objectives:

- *[Identify existing project information and the software components that should be tested.*
- *List the recommended Requirements for Test (high level).*
- *Recommend and describe the testing strategies to be employed.*
- *Identify the required resources and provide an estimate of the test efforts.*
- *List the deliverable elements of the test project.]*

1.2 Background

[Enter a brief description of the target-of-test (components, application, system, and so on) and its goals. Include information such as major functions and features, its architecture, and a brief history of the project. This section should only be about three to five paragraphs.]

1.3 Scope

[Describe the stages of testing—for example, Unit, Integration, or System—and the types of testing that will be addressed by this plan, such as Function or Performance.

Provide a brief list of the target-of-test's features and functions that will or will not be tested.

List any assumptions made during the development of this document that may impact the design, development or implementation of testing.

List any risks or contingencies that may affect the design, development or implementation of testing.

List any constraints that may affect the design, development or implementation of testing]

1.4 Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

[Note: Delete or add items as appropriate.]

Document (and version / date)	Created or Available	Received or Reviewed	Author or Resource	Notes
Requirements Specification	☐ Yes ☐ No	☐ Yes ☐ No		
Functional Specification	☐ Yes ☐ No	☐ Yes ☐ No		
Use-Case Reports	☐ Yes ☐ No	☐ Yes ☐ No		
Project Plan	☐ Yes ☐ No	☐ Yes ☐ No		
Design Specifications	☐ Yes ☐ No	☐ Yes ☐ No		
Prototype	☐ Yes ☐ No	☐ Yes ☐ No		
User's Manuals	☐ Yes ☐ No	☐ Yes ☐ No		
Business Model or Flow	☐ Yes ☐ No	☐ Yes ☐ No		
Data Model or Flow	☐ Yes ☐ No	☐ Yes ☐ No		
Business Functions and Rules	☐ Yes ☐ No	☐ Yes ☐ No		
Project or Business Risk Assessment	☐ Yes ☐ No	☐ Yes ☐ No		

2. Requirements for Test

The listing below identifies those items—use cases, functional requirements, and non-functional requirements—that have been identified as targets for testing. This list represents what will be tested.

[Enter a high level list of the major test requirements.]

3. Test Strategy

[The Test Strategy presents the recommended approach to the testing of the target-of-test. The previous section, Requirements for Test, described what will be tested—this describes how the target-of-test will be tested.

For each type of test, provide a description of the test and why it is being implemented and executed.

If a type of test will not be implemented and executed, indicate this in a sentence stating the test will not be implemented or executed and stating the justification, such as "This test will not be implemented or executed. This test is not appropriate."

The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

In addition to the considerations provided for each test below, testing should only be executed using known, controlled databases in secured environments.]

3.1 Testing Types

3.1.1 Data and Database Integrity Testing

*[The databases and the database processes should be tested as a subsystem within the <Project **Name**>. These subsystems should be tested without the target-of-test's User Interface as the interface to the data. Additional*

research into the DataBase Management System (DBMS) needs to be performed to identify the tools and techniques that may exist to support the testing identified below.]

Test Objective:	*[Ensure database access methods and processes function properly and without data corruption.]*
Technique:	• *[Invoke each database access method and process, seeding each with valid and invalid data or requests for data.* • *Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons]*
Completion Criteria:	*[All database access methods and processes function as designed and without any data corruption.]*
Special Considerations:	• *[Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases.* • *Processes should be invoked manually.* • *Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.]*

3.1.2 Function Testing

[Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions 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 Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the testing recommended for each application:]

Test Objective:	*[Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval.]*
Technique:	*[Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:* • *The expected results occur when valid data is used.* • *The appropriate error or warning messages are displayed when invalid data is used.* • *Each business rule is properly applied.]*
Completion Criteria:	• *[All planned tests have been executed.* • *All identified defects have been addressed.]*
Special Considerations:	*[Identify or describe those items or issues (internal or external) that impact the implementation and execution of function test]*

3.1.3 Business Cycle Testing

[Business Cycle Testing should emulate the activities performed on the <Project Name> over time. A period should be identified, such as one year, and transactions and activities that would occur during a year's period should be executed. This includes all daily, weekly, and monthly cycles and, events that are date-sensitive, such as ticklers.]

Test Objective	*[Ensure proper target-of-test and background processes function according to required business models and schedules.]*
Technique:	*[Testing will simulate several business cycles by performing the following:* • *The tests used for target-of-test's function testing will be modified or enhanced to increase the number of times each function is executed to simulate several different users over a specified period.* • *All time or date-sensitive functions will be executed using valid and invalid dates or time periods.* • *All functions that occur on a periodic schedule will be executed or launched at the appropriate time.* • *Testing will include using valid and invalid data to verify the following:* • *The expected results occur when valid data is used.* • *The appropriate error or warning messages are displayed when invalid data is used.* • *Each business rule is properly applied.*
Completion Criteria:	• *[All planned tests have been executed.* • *All identified defects have been addressed.}*
Special Considerations:	• *[System dates and events may require special support activities* • *Business model is required to identify appropriate test requirements and procedures.]*

3.1.4 User Interface Testing

[User Interface (UI) 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 target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.]

Test Objective:	*[Verify the following:* • *Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to-field, and use of access methods (tab keys, mouse movements, accelerator keys)* • *Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.]*
Technique:	*[Create or modify tests for each window to verify proper navigation and object states for each application window and objects.]*
Completion Criteria:	*[Each window successfully verified to remain consistent with benchmark version or within acceptable standard]*
Special Considerations:	*[Not all properties for custom and third party objects can be accessed.]*

3.1.5 Performance Profiling

[Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.

Note: Transactions below refer to "logical business transactions". These transactions are defined as specific use cases that an actor of the system is expected to perform using the target-of-test, such as add or modify a given contract.]

Test Objective:	*[Verify performance behaviors for designated transactions or business functions under the following conditions:* • *normal anticipated workload* • *anticipated worst case workload]*
Technique:	• *[Use Test Procedures developed for Function or Business Cycle Testing.* • *Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs.* • *Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).]*
Completion Criteria:	• *[Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction.]* • *[Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.]*
Special Considerations:	*[Comprehensive performance testing includes having a background workload on the server.* *There are several methods that can be used to perform this, including:* • *"Drive transactions" directly to the server, usually in the form of Structured Query Language (SQL) calls.* • *Create "virtual" user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with "traffic".* • *Use multiple physical clients, each running test scripts to place a load on the system.* *Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.* *The databases used for Performance Testing should be either actual size or scaled equally.]*

3.1.6 Load Testing

[Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and ability of the target-of-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, such as response times, transaction rates, and other time sensitive issues).]

[Note: Transactions below refer to "logical business transactions". These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

Test Objective:	*[Verify performance behavior time for designated transactions or business cases under varying workload conditions.]*
Technique:	• *[Use tests developed for Function or Business Cycle Testing.* • *Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.]*
Completion Criteria:	*[Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.]*
Special Considerations:	• *[Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.* • *The databases used for load testing should be either actual size or scaled equally.]*

3.1.7 Stress Testing

[Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle.]

[Note: References to transactions below refer to logical business transactions.]

Test Objective:	*[Verify that the target-of-test functions properly and without error under the following stress conditions:* • *little or no memory available on the server (RAM and DASD)* • *maximum actual or physically capable number of clients connected or simulated* • *multiple users performing the same transactions against the same data or accounts* • *worst case transaction volume or mix (see Performance Testing above).* *Notes: The goal of Stress Testing might also be stated as identify and document the conditions under which the system FAILS to continue functioning properly.* *Stress Testing of the client is described under section 3.1.11, Configuration Testing.]*
Technique:	• *[Use tests developed for Performance Profiling or Load Testing.* • *To test limited resources, tests should be run on a single machine, and RAM and DASD on server should be reduced or limited.* • *For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix.*
Completion Criteria:	*[All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions.]*
Special Considerations:	• *[Stressing the network may require network tools to load the network with messages or packets.* • *The DASD used for the system should temporarily be reduced to restrict the available space for the database to grow.* • *Synchronization of the simultaneous clients accessing of the same records or data accounts.]*

3.1.8 Volume Testing

[Volume Testing subjects the target-of-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 target-of-test can handle for a given period. For example, if the target-of-test is 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.]

Test Objective:	*[Verify that the target-of-test successfully functions under the following high volume scenarios:* • *Maximum (actual or physically- capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period.* • *Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.]*
Technique:	• *[Use tests developed for Performance Profiling or Load Testing.* • *Multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix (see Stress Testing above) for an extended period.* • *Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries and report transactions simultaneously for extended periods.]*
Completion Criteria:	• *[All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.]*
Special Considerations:	*[What period of time would be considered an acceptable time for high volume conditions, as noted above?]*

3.1.9 Security and Access Control Testing

[Security and Access Control Testing focus on two key areas of security:

- *Application-level security, including access to the Data or Business Functions*
- *System-level Security, including logging into or remote access to the system.*

Application-level security ensures that, based upon the desired security, actors are restricted to specific functions or 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 users granted access to the system are capable of accessing the applications and only through the appropriate gateways.]

Test Objective:	• Application-level Security: [*Verify that an actor can access only those functions or data for which their user type is provided permissions.]* • System-level Security: *Verify that only those actors with access to the system and applications are permitted to access them*.]
Technique:	• Application-level Security: [*Identify and list each user type and the functions or data each type has permissions for.]* • *[Create tests for each user type and verify each permission by creating transactions specific to each user type.]* • *Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied.* • System-level Access: *[See Special Considerations below]*
Completion Criteria:	*[For each known actor type the appropriate function or data are available, and all transactions function as expected and run in prior Application Function tests.]*
Special Considerations:	*[Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.]*

3.1.10 Failover and Recovery Testing

[Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with 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 Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.]

Test Objective:	*[Verify that recovery processes (manual or automated) properly restore the database, applications, and system to a desired, known, state. The following types of conditions are to be included in the testing:* • *power interruption to the client* • *power interruption to the server* • *communication interruption via network servers* • *interruption, communication, or power loss to DASD and or DASD controllers* • *incomplete cycles (data filter processes interrupted, data synchronization processes interrupted).* • *invalid database pointer or keys* • *invalid or corrupted data element in database]*

Technique:	*[Tests created for Function and Business Cycle testing should be used to create a series of transactions. Once the desired starting test point is reached, the following actions should be performed, or simulated, individually:* • *Power interruption to the client: power the PC down.* • *Power interruption to the server: simulate or initiate power down procedures for the server.* • *Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers.* • *Interruption, communication, or power loss to DASD and DASD controllers: simulate or physically eliminate communication with one or more DASD controllers or devices.* *Once the above conditions or simulated conditions are achieved, additional transactions should be executed and upon reaching this second test point state, recovery procedures should be invoked.* *Testing for incomplete cycles utilizes the same technique as described above except that the database processes themselves should be aborted or prematurely terminated.* *Testing for the following conditions requires that a known database state be achieved. Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.]*
Completion Criteria:	*[In all cases above, the application, database, and system should, upon completion of recovery procedures, return to a known, desirable state. This state includes data corruption limited to the known corrupted fields, pointers or keys, and reports indicating the processes or transactions that were not completed due to interruptions.]*
Special Considerations:	• *[Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required.* • *Resources from the Systems (or Computer Operations), Database, and Networking groups are required.* • *These tests should be run after hours or on an isolated machine.]*

3.1.11 Configuration Testing

[Configuration testing verifies the operation of the target-of-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—for example, applications, drivers, and so on—and at any one time, many different combinations may be active using different resources.]

Test Objective:	*[Verify that the target-of-test functions properly on the required hardware and software configurations.]*
Technique:	• *[Use Function Test scripts.* • *Open and close various non-target-of-test related software, such as the Microsoft applications, Excel and Word, either as part of the test or prior to the start of the test.* • *Execute selected transactions to simulate actor's interacting with the target-of-test and the non-target-of-test software.* • *Repeat the above process, minimizing the available conventional memory on the client workstation.]*
Completion Criteria:	*[For each combination of the target-of-test and non-target-of-test software, all transactions are successfully completed without failure.]*
Special Considerations:	• *[What non-target-of-test software is needed, is available, and is accessible on the desktop?* • *What applications are typically used?* • *What data are the applications running; for example, a large spreadsheet opened in Excel or a 100- page document in Word?* • *The entire systems, netware, network servers, databases, and so on also needs to be documented as part of this test.]*

3.1.12 Installation Testing

[Installation testing has two purposes. The first is to insure that the software can be installed under different conditions—such as a new installation, an upgrade, and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing.]

Test Objective:	*Verify that the target-of-test properly installs onto each required hardware configuration under the following conditions:* • *new installation, a new machine, never installed previously with <Project Name>* • *update, machine previously installed <Project Name>, same version* • *update, machine previously installed <Project Name>, older version*
Technique:	• *[Manually or develop automated scripts, to validate the condition of the target machine—new - <Project Name> never installed; <Project Name> same version or older version already installed).* • *Launch or perform installation.* • *Using a predetermined sub-set of function test scripts, run the transactions.]*
Completion Criteria:	*<Project Name> transactions execute successfully without failure.*
Special Considerations:	*[What <Project Name> transactions should be selected to comprise a confidence test that <Project Name> application has been successfully installed and no major software components are missing?]*

3.2 Tools

The following tools will be employed for this project:

[Note: Delete or add items as appropriate.]

	Tool	Vendor/In-house	Version
Test Management			
Defect Tracking			
ASQ Tool for functional testing			
ASQ Tool for performance testing			
Test Coverage Monitor or Profiler			
Project Management			
DBMS tools			

4. Resources

[This section presents the recommended resources for the <Project Name> project, their main responsibilities, and their knowledge or skill set.]

4.1 Roles

This table shows the staffing assumptions for the project.

[NOTE: Delete or add items as appropriate.]

Human Resources		
Worker	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Test Manager, Test Project Manager		Provides management oversight. Responsibilities: • provide technical direction • acquire appropriate resources • provide management reporting
Test Designer		Identifies, prioritizes, and implements test cases. Responsibilities: • generate test plan • generate test model • evaluate effectiveness of test effort
Tester		Executes the tests. Responsibilities: • execute tests • log results • recover from errors • document change requests
Test System Administrator		Ensures test environment and assets are managed and maintained. Responsibilities: • administer test management system • install and manage access to test systems

Database Administrator, Database Manager		Ensures test data (database) environment and assets are managed and maintained. Responsibilities: • administer test data (database)
Designer		Identifies and defines the operations, attributes, and associations of the test classes. Responsibilities: • identifies and defines the test classes • identifies and defines the test packages
Implementer		Implements and unit tests the test classes and test packages. Responsibilities: • creates the test classes and packages implemented in the test model

4.2 System

The following table sets forth the system resources for the testing project.

[The specific elements of the test system are not fully known at this time. It is recommended that the system simulate the production environment, scaling down the accesses and database sizes if and where appropriate.]

[Note: Delete or add items as appropriate.]

System Resources	
Resource	Name / Type
Database Server	
Network or Subnet	TBD
Server Name	TBD
Database Name	TBD
Client Test PC's	
Include special configuration requirements	TBD
Test Repository	
Network or Subnet	TBD
Server Name	TBD
Test Development PC's	TBD

5. Project Milestones

[Testing of <Project Name> should incorporate test activities for each of the test efforts identified in the previous sections. Separate project milestones should be identified to communicate project status accomplishments.]

Milestone Task	Effort	Start Date	End Date
Plan Test			
Design Test			
Implement Test			
Execute Test			
Evaluate Test			

6. Deliverables

[In this section, list the various documents, tools, and reports that will be created, by whom, delivered to who, and when delivered.]

6.1 Test Model

[This section identifies the reports that will be created and distributed from the test model. These artifacts in the test model need to be created or referenced in the ASQ tools.]

6.2 Test Logs

[Describe the method and tools used to record and report on the test results and testing status.]

6.3 Defect Reports

[In this section, identify the method and tools used to record, track, and report on test incidents and their status.]

Appendix A Project Tasks

Below are the test-related tasks:

- Plan Test
  - identify requirements for test
  - assess risk
  - develop test strategy
  - identify test resources
  - create schedule
  - generate Test Plan
- Design Test
  - prepare workload analysis
  - identify and describe test cases
  - identify and structure test procedures
  - review and assess test coverage
- Implement Test
  - record or program test scripts
  - identify test-specific functionality in the Design and Implementation Model
  - establish external data sets
- Execute Test
  - execute Test procedures
  - evaluate execution of Test
  - recover from halted Test
  - verify the results
  - investigate unexpected results
  - log defects
- Evaluate Test
  - evaluate Test-case coverage
  - evaluate code coverage
  - analyze defects
  - determine if Test Completion Criteria and Success Criteria have been achieved