# "Some Loan App" Release 1.0 Test Plan

## Overview

The “Some Loan App”, as it will be deployed in Release 1.0, allows Home Equity Loan Call Center Agents to fit home equity products (loans and lines of credit) to customers. At a high level, the system is configured as shown in figure one below. The “Some Loan App” itself is a group of Java programs and assorted “glue” that run on the WebLogic server. The Oracle database server provides storage as the application is processed, while the Netscape server offloads gateway activities to the clients from the WebLogic server. From a systems perspective, the “Some Loan App” can be said to “work” if it enables Call Center agents to initiate, negotiate, and close loan transactions with desirable customers while helping these agents avoid issuing loans to people unlike to pay them back.

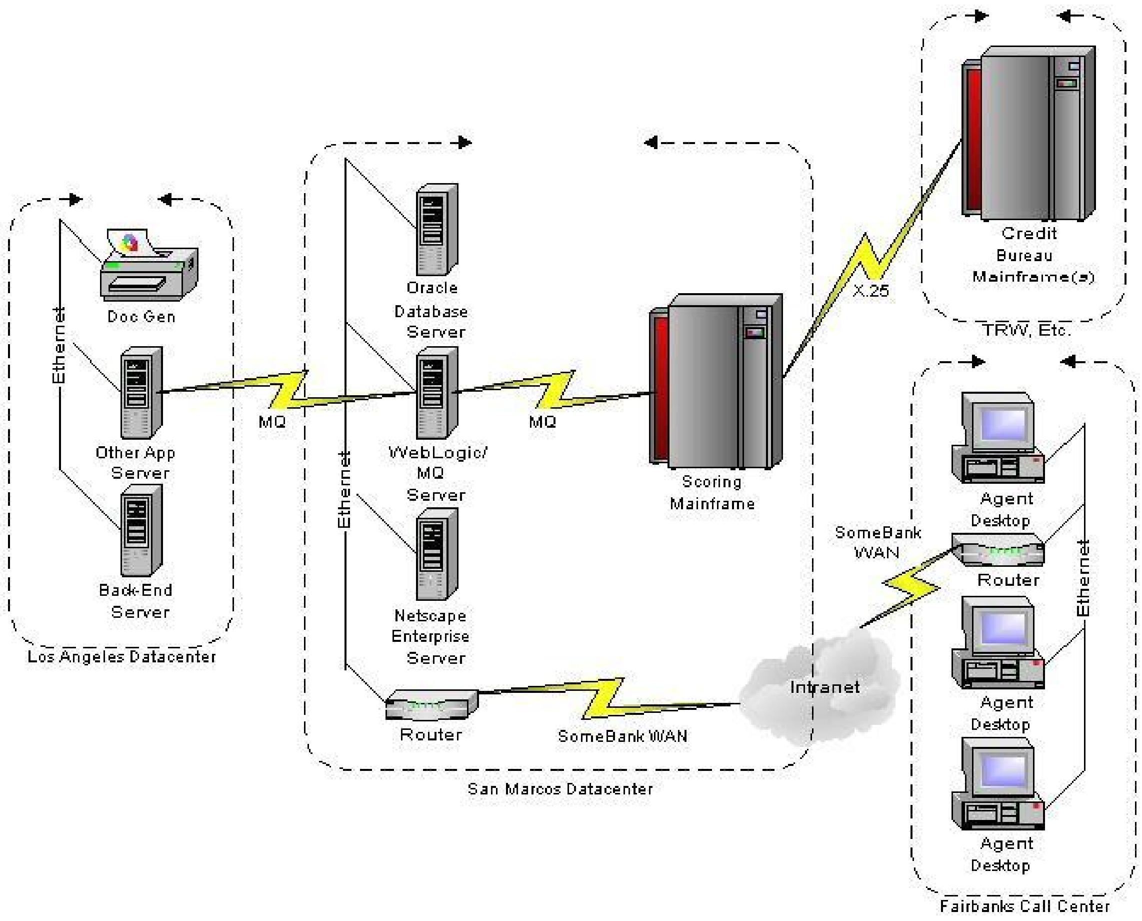

**Figure 1: “Some Loan App”, Release 1.0**

Loan transactions “flow” through the “Some Loan App” in the following fashion. Call Center agents in the Fairbanks Call Center receive phone calls from potential customers. They interview the customer, entering information into the “Some Loan App” through a Web browser interface. At a certain point, the “Some Loan App”, talking to the Scoring mainframe through MQ services, scores the customer’s credit. Based on this credit score, the "Some Loan App" displays various products that the Call Center agent can offer to the customer. If the customer chooses one of these products, the Call Center agent will conditionally offer the loan. The interview ends and "Some Loan App" transmits the loan information to "Some Other Loan App" for origination. Ultimately, though the services of the back-end server and documentation generator, the customer receives her loan documents.

The following test plan describes the testing to be performed by the Minneapolis Test Group (MTG), Somebank's independent test team for "Some Loan App" testing. This test plan covers the included items in the test project, the specific risks to product quality we intend to address, timeframes, the test environment, problems that could threaten the success of testing, test tools and harnesses we will need to develop, and the test execution process. Some testing occurs outside of the independent test team's area, such as unit testing. In general, our testing includes positive use cases developed by Robert S., Assistant Vice President, Home Equity Group, and capacity and performance test cases.

## Bounds

The following sections serve to frame the role of the independent test organization on this project.

### Scope

The following table defines the scope of the MTG independent test effort.

| MTG Independent Test (Release 1.0) | |
|---|---|
| **IS** | **IS NOT** |
| Positive use cases (functionality) | Negative use cases |
| Capacity and volume | Operations (e.g., paperwork processing, loan initiation, rate updates, etc.) |
| Error handling and recovery | Usability or user interface |
| Standards and regulatory compliance (as covered in the use cases) | Date and time processing |
| Client configuration (browser and call center desktop compatibility) | Localization |
| Security [TBD by Todd R. 7/22] | Test database development |
| Distributed (leverage Webdotbank testing) | Documentation |
| Performance | Code coverage |
| Black-box/behavioral testing | Software reliability |
| "Some Loan App"/"Some Other Loan App" status communications | Testing of the complete system |
| Confirmation testing in QA region | Horizontal (end-to-end) integration |
| | Data flow or data quality |
| | Unit or FVT testing |
| | White-box/structural testing |

**Table 1: MTG independent test team IS/IS NOT (scope)**

Clearly, the test team could do more, but we do not want to raise the quality bar for the "Some Loan App" for Release 1.0 at this stage in the game. Subsequent test efforts—i.e., Release 1.1 testing—will include many of the items in the "IS NOT" column above.

## Definitions

The following table defines commonly used test terms and other terms found in this document.

| **Term** | **Meaning** |
|---|---|
| Acceptance Test | A set of tests, usually selected from the System Test suite, chosen to demonstrate that the System Under Test (see below) meets reasonable quality standards. |
| Black Box Testing | Testing based on the purposes a program serves; i.e., behavioral testing. |
| Bug | Some aspect of the system under test that causes it to fail to meet reasonable expectations. "Reasonable" is defined by iterative consensus if it is not obvious. |
| Build | A collection of software objects of known revision levels compiled into one or more software executables for installation on the system under test. |
| Code Promotion | Installing a software build, in the form of a JAR file, into a server region. Also known as "build apply." |
| Confirmation Test | A selected set of tests designed to find ways in which a bug fix failed to address the reported problem fully. |
| Entry (Exit) Criteria | The parameters that determine whether one is ready to enter (exit) a test effort. |
| Integration Test | In this plan, a set of tests designed to find bugs in typical horizontal paths through integrated system components. |
| JAR File | Java archive containing a Java program. |
| Oracle | A method or procedure, often integrated into a test tool, for determining whether the system under test is behaving correctly. This can involve examining all the outputs or a sample of the outputs, either from a user perspective (behaviorally) or from a system internals perspective (structurally). |
| Quality Risk | The possibility of a specific system failure mode, either localized, caused by subsystem interactions, or a knock-on effect of a remote system failure, that adversely affects the system's user. |
| Reference Platform | A "known correct" system against which one can compare the results and behavior of the system under test. In this specific case, "Some Other Loan App" is the reference platform in terms of products offered, while the "Some Loan App" is the system under test. |
| Region | A set of servers configured to perform a particular role in terms of development, testing, or production. |

| Term | Meaning |
|---|---|
| Regression Test | A selected set of tests designed to find new failures, or regression, that changes, usually associated with bug fixes, have caused in subsystem, interface or product functionality. |
| Smoke Test | A limited set of regression tests (see above) designed to determine, though a random sample of critical functions, whether a given build is ready for testing. For this project, Smoke Test also includes confirmation tests (see above) in the Integration Region. |
| System Test | A set of tests designed to find bugs in the overall operation of the integrated system. |
| SUT | System under test. In this case, the Call Center agent desktop and browser, the HTML transmitted to the browser, the Netscape, the WebLogic, and the Oracle servers, the Scoring and "Some Other Loan App" interface modules, and the network infrastructure. |
| Test Escape | A field failure that could reasonably have been found by MTG executing this test plan but for errors in execution, attentiveness, interpretation of observed behavior, or other such problems. |
| Validation Test | A complete set of regression tests, including final confirmation of bug closure. |
| White-Box Testing | Testing based on the way a program performs its tasks; i.e., structural testing. |

**Table 2: Definitions**

### Setting

The test efforts described in this plan will take place in the following locations. (See Human Resources below for a description of the people in this table.)

| Location | Test Effort | Staff |
|---|---|---|
| "Some Client" (Minneapolis) | System test execution. | Minneapolis Test Group |
| RBCS (Bulverde) | Test project management. Test team staffing | Rex B. (off-site two days per week) |
| Webdotbank (Seattle) | Testing of the software components provided. | Webdotbank Test Team |

**Table 3: Locations involved in testing**

## Quality Risks

The test cases that will be used by the test team were primarily developed by the business in Fairbanks. As such, MTG doesn't have a lot of visibility into the quality risk management provided by them. MTG did develop test cases to cover specific bugs found during the Beta I process in Fairbanks; however, these do not correspond to specific quality risks, but rather to failure modes observed.

## Schedule of Milestones

The following shows the scheduled events that affect this test effort.

| **Milestone/Effort** | **Start** | **End** |
|---|---|---|
| Unit test complete | 4/22/98 | 4/22/98 |
| Smoke Build delivered and installed | 4/23/98 | - |
| System Test Entry Criteria Met | 4/24/98 | - |
| System Test (six release cycles) | 4/27/98 | 6/5/98 |
| System Test Launch Meeting | 4/27/98 | - |
| Build 1 delivered and installed | 4/27/98 | - |
| Build 2 delivered and installed | 5/4/98 | - |
| Build 3delivered and installed | 5/11/98 | - |
| Build 4 delivered and installed | 5/18/98 | - |
| Build 5 delivered and installed | 5/25/98 | - |
| Golden Code review (all bugs fixed: ready for final build) | 5/29/98 | - |
| Build 6 delivered and installed | 6/1/98 | - |
| System Test Exit Criteria Met | 6/5/98 | - |
| System Test Phase Exit Meeting | 6/5/98 | - |
| User Acceptance Test | 6/8/98 | 6/19/98 |
| Go/No-Go Decision | 6/22/98 | - |
| Deployment | 6/23/98 | 7/2/98 |

**Table 4: Scheduled System Test milestones**

## Transitions

The following subsections define the factors by which project management will decide whether we are ready to start, continue, and declare complete the test effort.

### System Test Entry Criteria

System Test can begin when the following criteria are met:

1. The “Tracker” bug tracking system is in place and available for all project participants.
2. All software objects are under formal, automated source code and configuration management control.
3. The HEG System Support team has configured the System Test clients and servers for testing, including the “cloned” call center agent desktops, the LoadRunner Virtual User hosts, the Netscape server, the WebLogic server, the Oracle server, the Oracle database tables (including indices and referential integrity constraints), the MQ connections, and the network infrastructure. The Test Team has been provided with appropriate access to these systems.
4. The Development Teams have code-completed all features and bug fixes scheduled for Release 1.0.

5. The Development Teams have unit-tested all features and bug fixes scheduled for Release 1.0, and transitioned the appropriate bug reports into a “verify” state.
6. Less than ten (10) must-fix bugs (as determined by Magdy J. , “Some Loan App” Release 1.0 Project Manager, and Robert S., Assistant Vice President, Home Equity Group) are open, including bugs found during unit testing.
7. The Development Teams provide revision-controlled, complete software products to MTG. (See the “Release Management” section.)

### System Test Continuation Criteria

System Test will continue provide the following criteria are met:

1. All software released to the Test Team is accompanied by Release Notes. These Release Notes must specify the bug reports the Development Teams believe are resolved in each software release.
2. No change is made to the “Some Loan App”, whether in source code, configuration files, or other setup instructions or processes, without an accompanying bug report.
3. Twice-weekly bug review meetings occur until System Test Phase Exit to manage the open bug backlog and bug closure times.

### System Test Exit Criteria

System Test will end when following criteria are met:

1. No panic, crash, halt, wedge, unexpected process termination, or other stoppage of processing has occurred on any server software or hardware for the previous three (3) weeks.
2. The Test Team has executed all the planned tests against the GA-candidate software release.
3. The Development Teams have resolved all must-fix bugs. Magdy J., “Some Loan App” Release 1.0 Project Manager, and Robert S., Assistant Vice President, Home Equity Group will define which bugs are must-fix.
4. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.
5. The open/close curve indicates that we have achieved product stability and reliability.
6. The Project Management Team agrees that the product, as defined during the final cycle of System Test, will satisfy the Call Center Agent’s reasonable expectations of quality.
7. The Project Management Team holds a System Test Phase Exit Meeting and agrees that these System Test exit criteria are met.

## Test Configurations and Environments

Testing involves both client systems and server configurations. There are two types of client systems present in the lab.

- Load Runner Virtual User clients (“LR clients”). These run Windows NT, have 128 MB of memory, and use Pentium 300 MHz CPUs. [Clarence: Please confirm configuration.] These systems are configured to support large numbers of simultaneous Load Runner “Virtual User” sessions.

- Call Center Desktop Agent clients ("CC clients"). These run Windows 95, have 64 MB of memory, and use Pentium 200 or better MHz CPUs. [Clarence: Please confirm configuration.] These systems are configured to resemble, as closely as possible in the "Some Client" network environment, the "Some Client Partner" Fairbanks Call Center Agent Desktop configuration.

The LR clients are used to run stress, performance, and capacity test cases, while the CC clients run the manual test cases. We have two LR clients in the test lab, two other "spare" LR clients under Clarence T.'s control, and six CC clients in the test lab.

These systems can connect to a variety of server configurations known as "regions." The following three regions are used during System Test.

- "Some Loan App" QA Region. This is the server environment into which the CC and LR clients send the "Some Loan App" loan applications during testing.
- Scoring QA Region. This server environment provides credit bureau scoring to the "Some Loan App" so it can decide which tier the customer suits the customer's credit risk.
- "Some Other Loan App" Regression Region. Once the tester has entered a test application into the "Some Loan App" in the QA region, the "Some Loan App" sends the loan to this "Some Other Loan App" server environment. After the "Some Loan App" sends the loan here, the tester will verify that the loan is present in the system and was correctly received by logging into "Some Other Loan App". He will then compare the correctness of the products offered by the "Some Loan App", by entering the same application into this "Some Other Loan App" server environment, then comparing the offered products.

Figure 2 below shows the various regions, how the tester interacts with them, and how they interact with each other.

The various regions used by the Development Team and MTG are accessed through the client system's Web browser. The browser is pointed to the correct region via a URL (favored approach) or an IP address (disfavored approach). The following table describes these access methods.

| Region | URL | IP |
|---|---|---|
| "Some Loan App" Integration | http://hegtest.bank.com | 168.118.19.205 |
| "Some Loan App" QA | http://hegqa.bank.com | 168.118.19.221 168.118.19.222 |

**Table 5: Regions used by Development and QA**

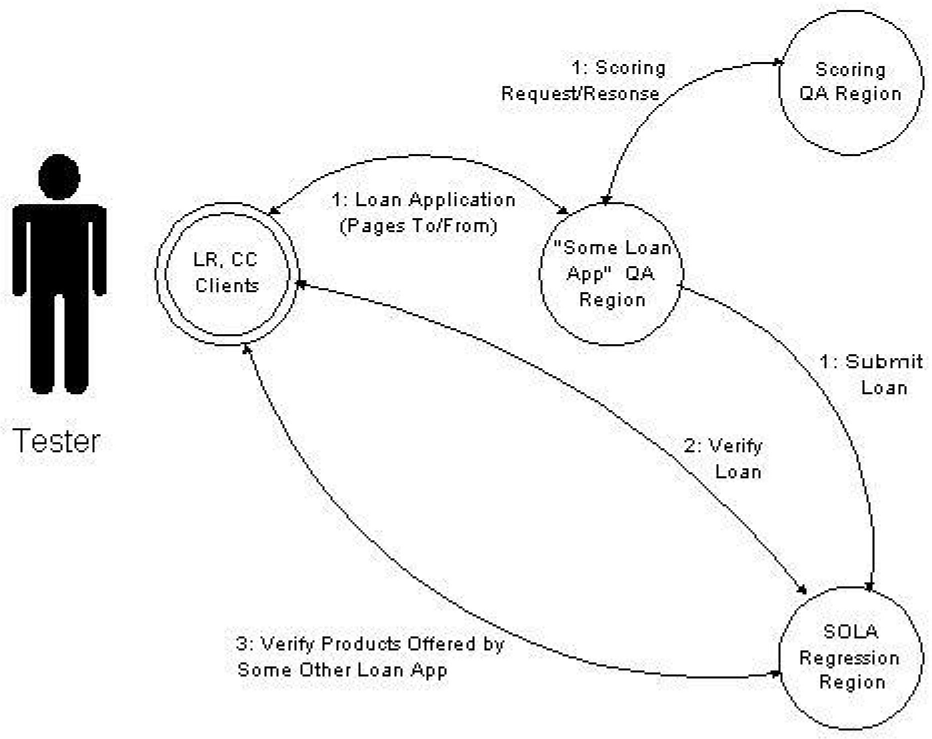
1: Scoring Request/Resonse
Scoring QA Region
Tester
1: Loan Application (Pages To/From)
LR, CC Clients
"Some Loan App" QA Region
1: Submit Loan
2: Verify Loan
3: Verify Products Offered by Some Other Loan App
SOLA Regression Region

**Figure 2: Regions used by testing**

As discussed elsewhere the Test Plan, the "Some Other Loan App" Regression Region serves as the reference platform. "Some Loan App" results, in terms of products offered, should match the results from "Some Other Loan App". Because of the handling of duplicate loan applications, "Some Other Loan App" will re-use the credit information provided to it by the "Some Loan App", meaning that the reference platform results are not entirely independent of the System Under Test results. It is conceivable that the "Some Loan App" could corrupt the scoring information in such as way as to cause "Some Other Loan App" to offer the same (wrong) products. If we have a test escape because of this risk, this problem will be caught in UAT when Call Center agents enter "live" applications into "Some Other Loan App". Finally, it is not possible to use "Some Other Loan App" as a reference platform in terms of performance, error handling, and so forth, because the behaviors of Unix character-based and Web GUI-based applications are so different.

## Test Development

The bulk of the manual testing performed for Release 1.0 consists of UAT test scripts developed by Robert S. and the Enormii Consulting Firm team as part of the original UAT/Integration Test effort. No new development is required, other than changes to these existing scripts to accommodate bug fixes required for defects found during the Beta One exercise in December.

Some of these bug fixes, however, required entire new areas of functionality. (These bugs were actually design changes rather than mere defects in the existing function.) We have implemented a number of new manual test cases to cover these changes. These test cases have been provided to Robert S. for review and approval.

Robert S. is also considering developing some additional tests to cover certain areas left untested by the original test scripts. If she does so, we will request these scripts in order to add them to our manual test suite.

Finally, Emma G. is working with Julius R. to redefine the automated Load Runner scripts that carry out the stress, capacity, and performance test cases. These efforts are, at this time, incremental; wholesale re-engineering of the automated suites will be part of the Release 1.1 effort. The most significant changes to the suites in terms of the tests performed arise from using more realistic time-per-page wait periods. These more-accurately model the time Call Center agents will spend filling in each page prior to submitting it to the "Some Loan App" servers.

## Test Execution

The following subsections define the activities and participants involved in test execution.

### Test Hours

MTG will test primarily between the hours of 9:00 AM and 5:00 PM (all times EST), Monday through Friday. However, we may start earlier and continue later, as the workload requires. The escalation process section below specifies on-hours and off-hours support numbers.

### Test Cycles

A test cycle will begin with each weekly release of a new software build as discussed elsewhere in this plan. Test execution will begin as soon as the new releases are appropriately installed. Testing will be carried out in the following order:

- **Confirmation testing.** The test team will retest all bugs reported as fixed in the release notes. For those bugs fixed, the testers change the state of the appropriate bug reports to "Resolved". They mark as "Not Resolved" reports for any bugs not fixed.[1]
- **Scheduled testing.** The test team will run all test cases scheduled for that test cycle.

Should the test team complete all scheduled test cases, it will perform any other test cases not scheduled for that cycle but which have not been run against the current build lab. If those cases are then completed, the test team will perform exploratory (ad hoc) testing until the following release begins the next cycle. Conversely, due to delays of deliverables, schedule pressures, high bug find rates, or the size (duration or effort) of the test suites, it may not be possible to complete all the scheduled test cases during every cycle. In that case, the test cases not completed will be rescheduled as first priority for the next cycle, immediately following the confirmation testing.

[1] In addition to testing performed by MTG in the QA Region, the Development Team will perform a daily confirmation test of the nightly builds provided by Webdotbank in the Integration Region. For the tests that fail, they will re-open the appropriate bug reports and return them for further work. MTG will not perform confirmation testing of these bug fixes since they have already failed in the Integration Region. For the Development Team tests that pass, they will re-assign the appropriate bug reports to Jenna B., who will coordinate the MTG confirmation testing in the QA Region.

## Test Execution Process

The objective of the System Test process is to find bugs in the system under test. We intend to do this by running a set of manual and automated test cases against each build. We want to run every test case at least one time, and ideally four times, as part of the System Test Phase. To do so, we will use the Test Case Summary worksheets discussed below. At the beginning of each test cycle, we will assign a "basket" of test cases to each tester. Each tester's basket will be different from one test cycle to the next, to ensure that any invalid assumptions on one tester's part do not result in a test escape. Once assigned their basket of tests, these testers will follow the process described in the rest of this section to execute each test case, and repeat the process until they have exhausted their list. If they empty their basket prior to the end of a cycle, they are to assist other testers by coordinating a reassignment of some of those testers' test cases to themselves. If all tests are completed, the test team will proceed as described in the section above.

Because testers must switch back and forth from the "Some Loan App" to "Some Other Loan App" to confirm correct product offerings, testers will run five scripts at once. The process is as follows:

- Log into the "Some Loan App" using personal ID, if it's the first batch of test cases for the day.
- Print five of the test scripts from the assigned basket.
- Enter the test scripts into the "Some Loan App" one at a time, recording the Fit Loan ID numbers as they appear.
- Record if the application is approved or declined. If one or more application is declined, and such are the expected results, the testers need not enter these applications into "Some Other Loan App".
- Log on to "Some Other Loan App" and find the loans transmitted by the "Some Loan App". Verify that they were transmitted correctly.
- Re-enter the identical test scripts into "Some Other Loan App".
- Record the "Some Other Loan App" APP number.
- Verify that the Credit Bureau response in "Some Other Loan App" matches the "Some Loan App"'s response, specifically if the loan is approved or declined.
- Verify the products offered in "Some Other Loan App" match the "Some Loan App" offerings, especially in terms of interest rate, terms and conditions, and amount.
- Prepare one or more bug reports immediately upon locating any discrepancies following the process described later in this Test Plan.
- Report status of the five test cases to Jenna B., Test Coordinator/Engineer, so that he can update the Test Case Summary spreadsheet for that cycle.
- Repeat this process for the next set of five test cases, or log out of the "Some Loan App" if this set was the last set for the day.

To ensure proper testing, the following parameters apply to this process:

- While testers will share "Some Other Loan App" login IDs, they will not "share" "Some Loan App" login IDs.
- Testers are expected to log out from the "Some Loan App" at the end of each day or before going on a break longer than half an hour.

- If a tester comes across an unattended CC client in the test lab logged into the "Some Loan App", the tester should log that client out of the "Some Loan App", especially if he or she plans to use that CC client to perform a test.
- Testers are to use **only** CC client systems for manual testing. Testers are not to use their personal desktops for manual testing, as they are not configured the same way as the Fairbanks Call Center Agent Desktops. LR clients are reserved for automated testing via Load Runner.

Automated testing proceeds as follows. Emma G., in consultation with Julius R., will finalize a set of performance and capacity test scripts. (See below for stepwise plan.) Clarence T. will then submit the LoadRunner jobs. (Clarence must be the "trigger man" because of "Some Client" IS Department concerns about errant LoadRunner processes flooding the production T1 that connects the Minneapolis office with the rest of "Some Client Partner".) Once the test runs are complete, Emma and Julius will analyze the results.

Based on previous findings, we do not expect the "Some Loan App" to satisfy the performance and capacity goals on the first run. Therefore, we will pursue a stepwise approach. We plan on five "steps." We hope to take one or more steps in each weekly cycle. Each step is a single type of test case. There are four other categories of test cases, based on accepted or declined applications and joint or single applications. Therefore, we have twenty performance test cases, as shown in the table below.

| | **Single** | | **Joint** | |
|---|---|---|---|---|
| **Simultaneous Users** | **Application Accepted** | **Application Declined** | **Application Accepted** | **Application Declined** |
| **20 Users** | Test Case 1 | Test Case 2 | Test Case 3 | Test Case 4 |
| **40 Users** | Test Case 5 | Test Case 6 | Test Case 7 | Test Case 8 |
| **60 Users** | Test Case 9 | Test Case 10 | Test Case 11 | Test Case 12 |
| **80 Users** | Test Case 13 | Test Case 14 | Test Case 15 | Test Case 16 |
| **100 Users** | Test Case 17 | Test Case 18 | Test Case 19 | Test Case 20 |

**Table 6: Stepwise approach to capacity and performance tests**

Because this testing, like the rest of the testing, must take place in the QA Region to be meaningful—this is the region that actually resembles the Production Region—Development cannot use this environment for debugging Performance problems by dropping in nightly builds once System Test proper begins. We recommend that the Development team use mathematical models to ensure that its performance-affecting design changes will enable sufficient performance before "burning" one of the four windows in which to run performance tests.

## Human Resources

The table describes the human resources need to execute this plan. Key support and liaison roles are also defined.

| Title | Roles | Name |
|---|---|---|
| Test Manager | Plan, track, and report on test execution<br>Secure appropriate human and other resources<br>Provide technical and managerial leadership of test team<br>Audit test processes and results of external test participants | Rex B. |
| Test Coordinator/ Engineer (Manual) | Develop manual tests<br>Execute manual tests<br>Train the other test engineers on manual tests<br>Track the test cases as discussed elsewhere in this plan | Jenna B. |
| Test Engineers (Manual) | Execute manual tests<br>Report results to the Test Coordinator | Hemamalini D.<br>Cheri L.<br>Greg J. |
| Test System Administrator | Ensure proper configuration of all the test lab workstations. | [TBH] |
| Test Toolsmith | Design and implement LoadRunner test suites<br>Interpret LoadRunner test results | Emma G. |
| HEG Level One Support | Provide first-line problem-escalation and test system configuration support to MTG for all regions involved in the System Test Effort<br>Escalate problems to “Some Loan App”, Scoring, “Some Other Loan App”, and SRM support as necessary (see process below). | Clarence T. |
| HEG Level One Support Manager | Manage the support and escalation processes | Mac D. |
| “Some Loan App” Manager | Coordinate with the Test Manager the “promotion” of weekly “Some Loan App” releases into the “Some Loan App” QA Region. | Hanna R. |
| “SOME OTHER LOAN APP” Release Manager | Coordinate with the “Some Loan App” Release Manager the “promotion” of as-needed “Some Other Loan App” changes into the “Some Other Loan App” Regression Region. | Sam C. |

| Title | Roles | Name |
|---|---|---|
| Scoring Release Manager | Coordinate with the "Some Loan App" Release Manager the "promotion" of as-needed Scoring changes into the Scoring QA Region. | Jenny V. |
| SRM Support | Deal with infrastructure issues in the QA Region. | Petra V. |
| SRM Build Manager | Promote the correct build weekly into the QA and Production Regions. | Petra V. |

**Table 7: Human resources for testing**

## Escalation Process

Unexpected failures, logistical challenges, and strange behavior occur during System Test efforts. Some of these events can block forward progress through the test suite. The following is the escalation plan when such blocking events occur during normal Test Hours. [Mac D./Magdy J.: Is this closed now?]

1. MTG shall escalate to Clarence T. any blocking issues encountered.
2. Clarence T., working with the appropriate Webdotbank, "Some Other Loan App", Scoring, and other resources, shall ascertain if the problem is an infrastructure issue. If so, he shall resolve the problem. (If not, see step four.) Status updates shall go out via e-mail at least every hour. At least the following people shall be on the distribution list, and, where these people have multiple e-mail addresses listed below, messages shall be sent to both addresses.
   - Magdy J.
   - Rex B.
   - Emma G.
   - Harry W.
   - Jenna B.
   - Hanna R.
3. Once Clarence has resolved the problem, he shall notify at least one member of the MTG via the phone numbers shown below that the problem is resolved, as well as sending out a final e-mail status update. (Steps four through seven are unnecessary if Clarence resolves the problem.)
4. If, after evaluation, Clarence concludes the infrastructure is functioning properly, thereby implicating the "Some Loan App" and its associated components as the root problem, Clarence shall hand off the problem to Jenny V. for resolution. Clarence shall pass the issue, his findings and the MTG contact person's name to Jenny V. He shall also inform participants of his conclusions and that Jenny V. has taken over the issue as follows:
   - The MTG contact person via the phone numbers shown below; and,
   - The e-mail distribution list shown above via e-mail.

5. Jenny V., working with the appropriate Webdotbank, “SOME OTHER LOAN APP”, Scoring, and other resources, shall resolve the problem in Fit, “SOME OTHER LOAN APP”, Scoring, or wherever it resides. Status updates shall go out via e-mail at least every hour. At least the following people shall be on the distribution list.
   - Magdy J.
   - Rex B.
   - Emma G.
   - Harry W.
   - Jenna B.
   - Hanna R.
6. Once Jenny V. has resolved the problem, he shall notify at least one member of the MTG via the phone numbers shown below that the problem is resolved, as well as sending out a final e-mail status update.
7. If Jenny V. feels it is appropriate, he shall instruct Clarence to rollback the code so that testing can continue. (This should only be done as a last resort.) Clarence shall inform Jenny V. upon completion of the rollback and Jenny V. shall close the issue as described in step six above.

During off-hours, the following process shall apply:

1. MTG shall escalate to the SRM duty phone any blocking issues encountered. The number left on the pager is the SRM contact number, so the MTG team member shall leave a cell phone number or a desk phone number with voice mail. (The MTG member shall only leave a voice mail phone number at which he or she can personally retrieve messages.)
2. Once MTG receives a return call from SRM, the MTG team member escalating the issue shall describe the problem, then exchange contact information for future follow up.
3. SRM shall provide regular updates on status, then an all-clear once the problem is resolved. If more than thirty minutes goes by without status, the MTG team member escalating the problem shall contact Rex B. via mobile phone. Rex B. may dismiss the staff for the remainder of the off-hours period—which may be the entire weekend—if SRM does not respond in a timely fashion.

Should any person on this list have difficulty, for whatever reason, executing the appropriate process, or should the process prove incapable of unblocking the problem within four hours, the MTG staff shall escalate the situation via mobile phone to Rex B. and Magdy J. Please note that all people named in the process below **shall** carry a mobile phone or pager, turned on with adequately charged batteries and sufficient coverage range, at all times during System Test. Automated and manual testing may—and probably will—occur continuously during System Test. Periods of unavailability shall be communicated to Magdy J. , “Some Client” “Some Loan App” Release 1.0 Project Manager, at least one week in advance, and the person making himself or herself unavailable shall arrange for alternate coverage.

## *Test Contact List*

| Name | Role | E-mail | Office | Cell/Pager |
|---|---|---|---|---|
| Hemamalini D. | Manual test engineer | hd@bank.com | (768) 555-1450 (lab) | N/A |
| Cheri L. | Manual test engineer | cl@bank.com | (768) 555-1450 (lab) | N/A |
| Greg J. | Manual test engineer | gj@bank.com | (768) 555-1450 (lab) | N/A |
| Jenna B. | Test coordinator/lead test engineer | jb@bank.com | (768) 555-1450 (lab) | (892) 555-1890 (c) (892) 555-0945 (p) |
| Emma G. | Test Toolsmith | eg@bank.com | N/A | (434) 555-6789 (c) |
| Rex B. | Test manager | Rex_Black @acm.org | (830) 438-4830 | (210) 241-8209 |

## *Support Contact List*

| Name | Role | E-mail | Office | Cell/Pager |
|---|---|---|---|---|
| Clarence T. | System administrator | ct@bank.com | (325) 555-1790 | (888) 555–1234 (c) (888) 555–9090 (p) |
| Jenny V. | System programmer | jv@bank.com | (267) 555-8903 | (800) 555-6892 (p) |
| SRM Hotline | SRM support | N/A | (142) 555-7894 | N/A |

## *Management Contact List*

| Name | Role | E-mail | Office | Cell/Pager |
|---|---|---|---|---|
| Magdy J. | "Some Loan App" R 1 Project Manager | mj@bank.com | (460) 555-8358 | (245) 555-9874 (c) (888) 555-9780 (p) |
| Mac D. | Support manager | md@bank.com | (465) 555-9834 | (465) 555-2308 (c) (888) 555-9024 (p) |
| Hanna R. | Webdotbank vendor manager | hr@bank.com | (372) 555-8010 | (215) 555-0945 (c) |

| Name | Role | E-mail | Office | Cell/Pager |
|---|---|---|---|---|
| Harry W. | Project manager | hw@bank.com | (456) 555-1345 | (222) 555-9085 (c) |

## Test Case and Bug Tracking

Test cases will be tracked using a set of Excel worksheets. These will provide both detail level and summary level information on test cases for each test pass. As discussed elsewhere, we intend to complete four passes through all the tests, so the Excel spreadsheet file will have eight worksheets, four pairs of Test Case Summary and Test Suite Summary worksheets. Jenna B., Test Coordinator/Engineer, will maintain and update these tracking documents. For each test case, we will track the information shown in the following table.

| Column Heading | Meaning |
|---|---|
| State | The state of the test case. The possible states are:<br>**Pass:** The test case concluded successfully.<br>**Warn:** The test case concluded with an error, which the project management team has either deferred, closed as external to the product, or closed as unavoidable.<br>**Fail:** The test case revealed a defect that development will address.<br>**Closed:** The test case previously revealed a failure that is now resolved.<br>**In Queue:** The test remains to be executed (indicated by a blank in the column).<br>**Skip:** The test will be skipped (explanation required in "Comment" column).<br>**Blocked:** The test cannot be run (explanation required in "Comment" column). |
| System Configurations | In most cases, the Workstation ID (from the front of the case) for the CC client where the tester entered the test case application, and the "Some Loan App" Region (usually QA) where the application was processed. |
| Bug ID | If the test failed, the identifier(s) assigned to the bug by Tracker when the tester entered the report. Jenna B. will enter each bug ID on a separate row below each test case row in the Test Case Summary worksheet. |
| Bug RPN | The risk priority number (severity times priority) of the bug(s), if applicable, in a column next to each bug ID. |
| Bug State | The state (see below) of the bug report, if any, filed against this test case. |
| Plan Hours | The planned effort (person-hours) for the first execution of this test case. |
| Actual Hours | The actual duration (person-hours) required. |

| Column Heading | Meaning |
|---|---|
| Plan Date | The planned date for the first execution of this test case. |
| Actual Date | The actual date it was first run. |
| Comment | Any comments related to the test case, required for those test cases in a “Skip” or “Blocked” state. |

**Table 8: Test tracking**

As the test organization runs each test, the state of each case will change from “In Queue” to one of the other states noted in the table above. In the ideal situation, at the System Test Phase Exit meeting, all the test cases will be in a “Pass” or “Closed” state on the Test Case Summary worksheet for the final pass.

For each test that identifies a problem and enters a “Fail” or “Warn” state, the tester will open a bug report in Tracker. For each defect, Tracker will store (at a minimum) the information shown in the following table.

| Field | Meaning |
|---|---|
| Bug ID | A unique identifier for each bug. |
| Title | A one- or two-sentence summary of the failure observed. |
| Failure Description | A text field, free-format, consisting of three sections:<br>**Steps to Reproduce:** A detailed, numbered process that will recreate the bug.<br>**Isolation:** The steps performed to isolate the problem, including verification on other platforms, bad-unit checking, and other pertinent tests.<br>**Regression:** A short description of whether this failure is a regression, and why or why not. |
| Severity | The absolute severity of the failure mode in question, without consideration of whether or not the failure mode is likely in operation, on a scale of 1 (worse) to 5 (least dangerous). The definitions are:<br>1. Loss of data.<br>2. Loss of functionality.<br>3. Loss of functionality with a workaround.<br>4. Partial loss of functionality.<br>5. Cosmetic error. |

| Field | Meaning |
|---|---|
| Priority | This is the priority, from a business perspective, of the failure mode. In other words, assuming we have this failure mode active or latent in our product on release, how badly will this affect our ability to use this product to assist "Some Client" in closing home equity loans and lines of credit over the entirety of the expected product life? Again, we use 1 (most dangerous) to 5 (least dangerous). The definitions are:<br>1. Defect renders system useless; must fix for ship.<br>2. Defect will unacceptably impact revenue; must fix for ship.<br>3. Fixing defect may not be as important as shipping product on schedule.<br>4. Release date more important; fix in next release.<br>5. No financial impact expected; defer or fix at convenience. |
| Resolution Notes | Once the bug is inactive (see Activity field below, "Closed"), this should include a description of the final resolution. |
| Submitter | The name of the tester or other engineer who identified the problem, defaulting to the current user. For remotely generated reports, this will specify either the contact name or the outsource test organization itself. |
| Submit Date | The date on which the bug report was opened. |
| Owner | The person responsible for moving the bug report to its next, and ultimately terminal, state. |
| Activity | To capture whether any action is pending on a bug, this field is either "Open" (further action is planned or needed at this time) or "Closed" (no further action will occur for Release 1.0). Open bug reports are in a "Review," "Reported," "Assigned," "Can't Reproduce," "Test," or "Not Resolved" state, while closed bug reports are in a "Rejected," "Defer" or "Resolved" state. |

| **Field** | **Meaning** |
|---|---|
| State | The state of the issue, as follows:<br>**Review:** Awaiting a peer review by another tester.<br>**Rejected:** Review failed; behavior is not a bug.<br>**Reported:** The problem is deemed by the test engineer fully characterized and isolated.<br>**Assigned:** The problem is accepted as fully characterized and isolated by development, and an owner, responsible for fixing the problem, is assigned.<br>**Test:** Development have repaired the problem in some level of hardware or software, and someone owns testing the problem to evaluate the fix.<br>**Not Resolved:** The fix failed the retest.<br>**Defer:** Do not fix for this release.<br>**Resolved:** The fix passed the retest. |
| Symptom | A classification of the symptom type as follows:<br>**Functionality.**<br>**Performance.**<br>**Usability.**<br>**Reliability.**<br>**Operations.**<br>**Compliance.**<br>**Design.** |
| Subsystem Affected | A classification of the subsystem most impacted by the bug as follows:<br>**User Interface.**<br>**Bank Interface.**<br>**"Some Other Loan App".**<br>**Scoring.**<br>**Other.** |
| Release Status | For old bug tracking database compatibility, from the following list:<br>**NDE, High.**<br>**NDE, Low.**<br>**IFP, High.**<br>**IFP, Low.**<br>**Deferred.** |

| Field | Meaning |
|---|---|
| Must Fix For Phase | A target phase for bug resolution from the following list:<br>**Unit Test.**<br>**System Test.**<br>**UAT.**<br>**Deployment.**<br>**Beta.**<br>**Production.**<br>**Next Release.**<br>**N/A.** |
| Test ID | The test identifier (from the test-tracking spreadsheet described above) corresponding to the test case the engineer ran that uncovered the issue. Also allowed are “Ad Hoc”, “Other” and “Unknown”. |
| Opened Count | A numeric field incremented each time a bug is reopened, to gauge the extent of regression. |
| Version Number | The version of “Some Loan App” against which the bug was identified. (See “Release Management” section for a discussion of version numbers.) |
| Closed Date | The date on which the issue was confirmed fixed or put on hold, which is used only when the issue is in a “closed” or “deferred” state. |
| SIR ID Number | If the bug was identified and reported using the old bug tracking database, this is the old bug ID number for cross-reference. |

**Table 9: Bug tracking**

In order to ensure high-quality bug reports, the test team shall use the following process for reporting bugs.

1. Structure. Testers shall use a deliberate, careful approach to testing, and take careful notes. These notes form the basis of the failure description.
2. Reproduce. Testers shall check reproducibility of a failure before writing a bug report. The steps to reproduce shall appear in the failure description as discussed above. Sporadic failures shall be indicated by the keyword “intermittent” in the summary and the failure description fields.
3. Isolate. Testers shall isolate defects by changing certain variables, including the CC client tested, that may alter the symptom of the failure. The isolation attempts shall be documented in the failure description fields
4. Generalize. After the tester has an isolated and reproducible case, he shall investigate more general instances of the problem. Does the same failure occur in other modules or locations? Can he find more severe occurrences of the same fault?
5. Compare. If the test case or test condition exposing the bug was run previously, the tester shall indicate in the failure description field whether the bug is a regression. Also, if the failure involves an invalid product offering, the test shall indicate in the

isolation section of the failure description field whether "Some Other Loan App" exhibits the same bug.

6. Summarize. The tester shall write a one-line executive summary for each bug report, in non-technical language, that explains how the failure observed will or might affect the Call Center agent and/or Somebank's business.
7. Condense. The tester shall eliminate extraneous steps or words.
8. Disambiguate. The tester shall ensure that readers will not misinterpret the report due to words or phrases that are vague, misleading, or subjective.
9. Neutralize. The tester shall use only neutral, objective wording in their bug reports and confine their statements to those of facts.
10. Review. The tester shall submit his or her bug report for peer review prior to submitting them to the Development and Project Management teams.

The test team shall write an accurate, concise, thoroughly-edited, well-conceived, high-quality bug reports. Comments, questions, or concerns about this process or the quality of the bug reports coming from MTG during System Test should be directed to the Rex B.

To avoid test escapes, the testers shall adopt **an active bias towards bug reporting**. This implies the following attitudes and behaviors:

- The Test Manager shall inform the team that its role is to find bugs, and shall reinforce this message.
- If in doubt, testers are to assume the observed "Some Loan App" behavior is incorrect until they satisfy themselves otherwise.
- If "Some Other Loan App" and the "Some Loan App" disagree about the correct product offering, a bug exists and shall be reported.
- If the specifications, on-screen help, or any other official document and the "Some Loan App" disagree about correct behavior, a bug exists and shall be reported.
- The tester shall report as a bug any event leading to loss of data or the crash, wedge, hang, or other availability incident involving any server or network.
- If, in the professional opinion of the tester, the behavior of the "Some Loan App" does not conform to reasonable expectations of Web and/or GUI program behavior and/or quality, or is otherwise confusing, misleading, or ambiguous, the tester shall report a bug.
- Questions about which component a bug resides in do not affect whether anomalous behavior is incorrect. Testers shall report such bugs regardless of whether the assignment of responsibility for repair is clear.
- Disagreements between MTG and any other group about whether a behavior is incorrect shall be escalated to Magdy J. or Robert S. for resolution. They shall have the sole authority to cancel or defer any bug report for whatever reason.

## Release Management

During System Test, MTG will receive "Some Loan App" builds every Monday morning from the Development teams. MTG does not anticipate any need for new "Some Other Loan App" or Scoring builds into their respective Regression and QA regions, but, if such occur, they will occur in sync with a "Some Loan App" build delivery. MTG will run the entire set of tests against these builds. See Table 10 below for the build delivery and test schedule.

| Build Delivery Date | Test Cycle/Test Pass | Tests Run |
|---|---|---|
| 4/23/98 | Smoke | All |
| 4/27/98 | 1 | All |
| 5/4/98 | 2 | All |
| 5/11/98 | 3 (including agents) | All (plus agent exploratory) |
| 5/18/98 | 4 (including agents) | All (plus agent exploratory) |
| 5/25/98 | 5 (including. Agents) | All (plus agent exploratory) |
| 6/1/98 | 6/Validation | All |

**Table 10: Release schedule and test cycles**

The following five items are important for MTG in terms of release management.

1. **Predictability and timing of releases.** Releases that show up at unpredictable times, too frequently, or too rarely can impede the forward progress of testing. Releases are scheduled to move into the QA Region every Monday morning, with test launch at 10:00 AM. [Magdy J.: Can we convince SRM to support this schedule?] (When it is necessary to accept an out-of-cycle release to keep testing moving, test will do so.) Builds are moved into the Integration Region daily.

2. **Update apply process.** Ideally, the process of moving to a new release requires no external support, is simple, and is executable (via automation) by the Development teams in a matter of minutes. The specifics of this process for the "Some Loan App" in the Integration and QA Regions are discussed below.

3. **Update unapply process.** Sometimes bug fixes create more problems than they resolve, in which case a process for removing the update or recovering from the change is needed. Like the update process, simplicity and automation are goals. The specifics of this process for the "Some Loan App" in the Integration and QA Regions are discussed below.

4. **Naming.** When the test team reports new bugs, they need a way of identifying the offending releases. This requires a consistent naming convention for any subsystem release to test. The naming convention need not be meaningful, but it should imply sequence, as in A, B, C… For the "Some Loan App", the releases imply both sequence and date. They are of the form `SLA.MM.DD.YY.rN`, where `MM.DD.YY` is the date of the build, and `N` is the build number for the day; e.g., `SLA.03.07.00.r1`.

5. **Interrogation.** Naming conventions do very little good unless a tester has a way of determining the release names of all the subsystems by interrogating the system under

test. This process needs to be as simple as possible to enable quick reporting of bugs. [Hanna R.: Please provide method.]

When the Development teams provide a build to the test organization, they will also provide release notes. These release notes will, at a minimum, identify, by Bug ID, all bugs fixed in that release. A bug is considered "fixed" from a development standpoint when a developer has expended effort to resolve it and when he has run a unit test to confirm that the problem is indeed resolved. The release notes shall be provided as a text attachment to a release notification e-mail.
The release process to the Integration and QA Regions is as follows:

1. Webdotbank develops on Webdotbank-owned boxes.
2. Webdotbank performs pre-integration testing in Pre-Integration Region (hegdbqa2 - 224).
3. Some Loan App Development team develops on Developer's workstations.
4. Some Loan App team performs Integration Testing in the Train Region.
5. Webdotbank (Max J./Jacques F.) shall perform a build each evening. (If there is no build necessary for that day, an e-mail shall be sent to the parties list below informing them of this.) Each build shall contain complete version information and file naming, as well as a reference to the bug(s) fixed within that build and any associated DB script changes or similar documentation. Each build shall also be supplied with release notes as described above. Once the build is complete, Webdotbank shall notify the following parties that a build has been completed and is ready for promotion to Integration:
   - Hanna R.
   - Jenny V.
   - Clarence T.
   - Rex B.
   - Jenna B.
   - Emma G.
   - Magdy J.
6. At 9:00 AM each weekday, the Build Manager (Clarence T.) shall promote the nightly build to the Integration Region. If any Oracle database changes are necessary, Clarence shall work with Minnie P. in SRM (our DBA) to make those changes.
7. The Development Team shall perform Smoke Testing (limited Regression Testing) in the Integration Region.
8. Friday afternoon, Clarence T. shall inform the "Some Client" SRM contact, Petra V., of the revision of the "Some Loan App" to be promoted to the QA Region (nclqa.fusa.com ) and Production Region (hegprod.bank.com ) on Sunday. On Monday morning, Clarence T. shall confirm that the build was successfully promoted, then shall notify the following people via e-mail:
   - Magdy J.
   - Rex B.
   - Emma G.
   - Harry W.

- Jenna B.
- Hanna R.

9. Should the quality of the new build fail to meet the continuation criteria described above, the Test Manager shall halt testing and ask the "Some Loan App" Release Manager to perform an "unapply" operation, reverting to the previous release. Testing shall resume against that release, while waiting for an out-of-cycle build to repair the blocking problems.

The Scoring QA Region and "Some Other Loan App" Regression Region shall not change during "Some Loan App" System Test, except as the result of bugs found in Scoring or "Some Other Loan App" during "Some Loan App" System Test. If bug fixes or other code changes must occur in these regions to address such bugs, then the "Some Other Loan App" Release Manager and Scoring Release Manager shall coordinate with the "Some Loan App" Release Manager to ensure that all necessary code changes are applied to all three regions ("Some Loan App" QA Region, Scoring QA Region, and "Some Other Loan App" Regression Region) as part of the weekly build install.

## Risks and Contingencies

The following table describes the key risks to success of this plan, and contingencies to address them.

| Risk | Contingency |
|---|---|
| Insufficient test lab hardware. | Consider a day and a night shift, weekend hours, and so forth. |
| Tracker not available. | Test team enters all bugs by hand. This is a very inefficient and undesirable approach; I estimate a 10 to 25 percent hit. |
| Release management not well-defined, resulting in a test cycle's results being invalidated. | Define a crisp release management process.<br>Weekly builds into the QA Region only during System Test proper. |
| QA Region not ready for testing, leading to results not necessarily representative of the Production Region's expected behavior. | Start testing (functional only) in the Integration Region until the QA Region is ready.<br>Slip the System Test schedule day-for-day until the QA Region is available. |
| Test lab setup delayed or incomplete. | We can begin testing if the cloning is not done, but, if we must function with insufficient hardware (less than described in this plan), we will suffer a schedule impact. |
| Test environment system administration support not available or proficient. | Identify system administration resources with pager/cell 24x7availability and appropriate Unix, NT, and network skills. |

| Risk | Contingency |
|---|---|
| Buggy deliverables impede testing progress. | Complete unit testing by Webdotbank. Adherence to test entry and exit criteria. |
| Test and/or product features, quality, or schedule plans change, causing a rethinking of the test project. | Change management or change control board. Expansions in test system scope may necessitate "splitting" the test passes over multiple cycles. |
| "Some Other Loan App" or Scoring change without coordination with "Some Loan App" releases. | The "Some Loan App" is closely tied to these systems, so such changes reset the current test cycle and require starting over. A one-week slip in the System Test Phase Exit date should be expected should such an event occur. |

**Table 11: Risks and contingencies**

## Change History

The following table outlines the change history for this document.

| Revision | Released | Description/Changes | Author/Editor |
|---|---|---|---|
| 0.1 | 3/26/98 | First draft with questions and feedback requests to "Some Client" team. | Rex B. |
| 0.2 | 3/29/98 | Second draft with responses to some of the questions incorporated, sent to a wider audience. | Rex B. |
| 0.3 | 4/1/98 | Minor changes based on feedback from test team. | Rex B. |
| 0.9 | 4/10/98 | Incorporated comments from the plan review. Remaining questions escalated for resolution. | Rex B. |
| 1.0 | TBD | Final | Rex B. |

**Table 12: Change history**

## Referenced Documents

See the various documents in "Some\Network\Repository" for more information on how Release 1.0 of the "Some Loan App" is supposed to work.
See Julis R.'s performance document for more information on performance testing.
This test plan format complies with ANSI/IEEE Standard 829.

## Frequently Asked Question

The following section addresses some frequently asked questions, and should be kept as a reference for MTG team members and those who interface with them.

- What do I do if I get stuck and can't run tests?
  Good question. If you are stuck because you can't figure out what the "Some Loan App" should be doing, and you have checked "Some Other Loan App"'s behavior already, contact Hanna R. and Jenna B. If you are stuck because the infrastructure is misbehaving, contact Clarence T. or SRM as specified in the "Escalation Process" section.

- How do I get in touch with people when I'm stuck?
  The escalation and release processes described in this test plan are the normal channels to be tried first. Phone numbers are included in the contact lists. *Try normal channels before escalating outside the process.* If you can't get the issue resolved through normal channels *for whatever reason*, call Rex B. on his mobile phone at +1 (768) 555-1567. If he doesn't answer, call him at his home office at +1 (176) 555-1730. If you still can't get Rex, call Magdy J. on his mobile phone at +1 (779) 555-1540, or, if you can't reach her there, use her pager, +1 (888) 555-1984. DO NOT STAY STUCK. DO NOT GIVE UP. CONTINUE TO ESCALATE UNTIL SOMEONE TAKES OWNERSHIP OF GETTING YOU UNSTUCK.

- Do I have to file a bug report every time I encounter a problem?
  Yes, usually. If a bug report has already been written—by you or by someone else—you should add a note. If the problem is an infrastructure problem—e.g., the QA Region is very slow, a workstation is crashing, etc.—file a bug report *and then* proceed as described in the "Escalation Process" section.

- Do I need to have all my bug reports reviewed?
  Yes. All of them. Every single one. No bug report is to be formally submitted without a review. If you can't find anyone, call Rex B. on his cell phone and read the bug report to him.

- What if I'm confused about how PVCS Tracker works?
  Emma G. is the resident expert and the informal Tracker support resource for MTG. Ask him.

- How do I know what version of the "Some Loan App" is running in which region?
  Ideally, you could ask the "Some Loan App", analogous to the "Help/About" options in Windows applications. Right now, you can't, so see the paper taped to the left wall (the one with the Tracker workstations against it).

- When do new builds show up?
  The QA Region is to receive a new build every Monday morning from SRM. This process is still in flux, so Jenna B. will need to drive the process manually until it begins to work regularly. The Integration Region is to receive a new build every morning, given that Webdotbank has made a nightly build.

- How do I know whether my bugs are fixed in the new build?
  The release notes for each nightly build specify the bugs fixed in that build. So, in the QA Region, the complete set of release notes for all builds since the last build installed let you know which bugs are fixed. As a practical matter, Jenna B. will print a report every Monday for the bugs that are assigned to him and a "Test" state, ready for confirmation testing in the QA Region. The appropriate MTG team member or Developer will test each bug.

[Jenna B./Emma G., what do I need to add here?]