Michael Fiddes, William Papsco, Ashley Maddix, Zayne Stites, Colton Russell, Marina Neumann

Term Project: *ChocAn*

Test Plan Document

Table of Contents

1 Introduction **2**

1.1 Purpose and Scope 2

1.2 Target Audience 2

1.3 Terms and Definitions 2

2 Test Plan Description **2**

2.1 Scope of Testing 3

2.2 Testing Schedule 3

2.3 Release Criteria 3

3 Unit Testing **4**

3.1 Strategy 4

4 Smoke Testing **5**

4.1 Login Smoke Test 5

4.2 Provider Smoke Test 5

4.3 Manager Smoke Test 6

5 System Testing **7**

5.1 Login System Test 8

5.2 Provider System Test 8

5.3 Manager System Test 9

1 Introduction

In the process of developing a software product, it's critical that various testing is performed in order to ensure the quality of the program. This testing is done in different stages of the development process, stating by looking at each function individually and testing that it works and behaves as expected on its own. Moving from there, testing is done on the software once code has been integrated with each other. At this stage, the development team is verifying that these functions interact with one another as they should. At the last stage of testing before it can be determined ready for clients, the system is tested as a whole to ensure all the correct paths are taken and that the product is usable and of quality. In this document we outline the test plan for ChocAn's software by specifying the functionality to be tested within each stage and the expectations that should be met from these tests.

1.1 Purpose and Scope

This document outlines the testing strategy ICNH intends to use to test the data processing software being developed for the ChocAn system. This is crucial for demonstrating to the client that the software meets the functionality agreed upon in the requirements document, and does so accurately and without erroneous or erratic behavior.

1.2 Target Audience

This document is intended to help development teams test that the software meets the necessary level of functionality and dependability. It also provides the client assurance that the software is dependable to the extent described in this document.

1.3 Terms and Definitions

Happy path - A default use scenario containing no exceptions or error conditions.

Sad path - A use scenario containing exceptions and error conditions.

Smoke test - Tests that ensure happy path functionality is maintained throughout development.

Profile - A record dedicated to a single provider or member that contains all necessary information about that person or organization.

2 Test Plan Description

This section provides an overview of the testing strategies employed. It will generally cover what type of testing will be used on different sections of the software, as well as the schedule the testing will be performed on and the allowed degree of fault in test results.

2.1 Scope of Testing

Unit testing will be used to test the Login and each feature of the Provider and Manager Interface. All happy and sad paths outlined in the use cases section of the requirements document will be tested. Any feature that requires users to input information must be defect tested to ensure the appropriate behavior when a user enters invalid information. Each interface will be component tested to verify the interactions of the features work as expected. The database and database interfaces will be tested as part of the smoke and system tests.

2.2 Testing Schedule

Testing will immediately follow any changes to the implementation of each feature and component. Smoke and system testing will not begin until the four interfaces are implemented, but will be repeated with any updates to the system before the changes are pushed.

2.3 Release Criteria

As users will interact with the software exclusively through the terminals, only input from an extended QWERTY keyboard will be tested (i.e. arabic numerals, english alphabet, alt num-pad characters). When all tests pass under these conditions, the test results can be shared with the client for negotiation, following which the software is considered ready to be released.

3 Unit Testing

In the Login Interface, the user ID entry will have unit tests for valid and various forms of invalid ID (i.e. non-numeric entry, null entry, input larger than buffer).

In the Provider Interface, the following features will have at least one unit test per feature: check member ID status, charge member, provider directory lookup. Each of these will have the standard input validity unit tests, but the features will not be fully tested as they require component interactions thus fall into the integration test category.

In the Manager Interface, the following features will have at least one unit test per feature: provider report, member report, add member, remove member, update member, add provider, remove provider, update provider. The unit tests for these features will be limited to input validity tests, as, like the provider interface features, further testing falls into the integration test category.

The database interface will have no unit testing, as unit testing is inappropriate and would lead to a decrease in database quality.

3.1 Strategy

The target is 80% code coverage for unit testing. The various databases will not be tested and makeup approximately 15% of the code base. The remaining 5% is an estimate of the features which aren't suitable for unit testing, i.e. database retrieval features in the provider interface. Of the target 80%, the vast majority of unit testing will cover user input validation. This is suitable because the main purpose of the software is to process data for ChocAn according to user input, so it is assumed that once the user input has been validated and passed all unit tests, changes to the database as a result of that input is correct and intended from a unit test standpoint. The integrity of the database interface will be tested by proxy as part of the system and smoke tests that will take place after each interface has been implemented. For the system tests to pass, the database must accurately store and report data as intended. Therefore if the system tests pass, the database is reliable and accurate.

4 Smoke Testing

Once the Software is in running order the system is then tested to ensure its stability. These tests determine that no major defects occur while navigating and using the product according to the “happy path”. While in this testing stage, software stopping errors and many major defects may be discovered. It’s important to run these tests with every new build of the software to ensure it retains its usability, and to check that no major errors are introduced when porting the software to the ChocAn terminals upon delivery. These tests will not be automated as we want to verify that users can still interface with the software as expected, not just call and test the underlying methods.

4.1 Login Smoke Test

In this software, the core data processing functionality is granted to specific users only after they have logged in. This means that the login system is critical to the usability of this program. The smoke test for this functionality ensures that we get expected behavior from a valid input. A tester will run the program and attempt to login with a valid user ID. Once a valid ID has been entered, it is expected that the login attempt will succeed, access to the appropriate functionality will be granted to the user, and the test then passes. If the login attempt fails with a valid key, the test fails and there exists a critical error in the system indicative of a potentially greater problem.

4.2 Provider Smoke Test

Providers need to be able to perform multiple operations: check a member’s status, log a given service, look up a service in the provider directory, and compile a report for services provided. This smoke test will assert that each of these operations is correctly executed for a valid input. A tester will login as a provider, choose to check the status of a specific member, and input a valid member ID. It is expected that the function will return an active status for that member, and if so the test passes.

To test that a provider can correctly log a given service, a tester will login as a provider, choose to log a service, enter a valid member ID, and enter in the relevant information for the service provided. It is expected that the service will be logged to the database with no issues.

To test correct access to the provider directory, a tester will once again login as a provider, and choose to lookup a service in the directory. The provider will enter the name of a service in the directory, and it is expected that the service will be found in the database and the appropriate information printed to the screen.

To test that a provider can request and receive a report for services provided, a tester will login as a provider, log a service to the database, then request the weekly report. It is expected that the software will output a text file containing the information for the service previously logged.

4.3 Manager Smoke Test

As with providers, managers require the ability to perform many tasks. These tasks are: compiling a service report for providers, compiling a service report for members, and adding, removing, and updating members/providers.

The smoke test for both reports follow a similar path for that of the provider test. A tester will login as a provider, log some service to the database, logout, login as a manager, and request a service report for each provider. It is expected that the software will output a text file for the provider the service was logged under, containing information about the service logged. The tester will then request a service report for each member. It is expected that the software will output a text file for the member the service was logged to, containing information about the service logged. If there are pre-existing services logged in the system, the reports will contain information about all services logged in the last week for the provider or member in question.

To test that a member can be added, removed, and updated, a tester will login as a manager, and choose to add a member. The tester will fill in the necessary information asked of them, set the member status to active/valid, and submit the request. The tester will then logout, login as a provider, and attempt to log a service to the new member. It is expected that the service will be correctly logged to the database. Following this, the tester will relogin as a manager, and update a member’s information, inputting the recently added member’s ID. The tester will change the

member's address, and submit the change. After this, the tester will request a service report for the member. It is expected that the report will reflect the new address for the member. Proceeding to the last member test, the tester will attempt to remove a member, inputting the recently added member's ID. It is expected that a final report will be generated for the member if they have any services logged, then the member and any services logged to the member will be removed from the database. The tester will attempt to compile a report for the member. It is expected that the attempt will fail as the member cannot be found.

To test that a manager can add, remove, and update providers, the tester will closely follow the steps followed for adding, removing, and updating member information, but this time with a provider. The tester may need to add a valid member to the system to test that the new provider can log services correctly, and reports need to be compiled for the provider rather than the member, but otherwise these tests match those described previously.

Once these tests have passed, we are relatively sure that the core functionality of the software has been maintained across updates and/or platforms.

5 System Testing

The system tests need to demonstrate that the system correctly handles all foreseeable paths through the system. This means that happy paths, sad paths, and combinations of both paths are handled without error or negative impact on system integrity. The smoke tests are a subset of the system tests, so these system tests will closely resemble the smoke tests for each system but with a focus on error checking. Some tests may be automated, and some may be performed by a live tester.

5.1 Login System Test

The login system has minimal parts, as it acts simply as a gate to the main functionality of the software. Therefore the system tests for the login system are rather few. From the main login menu, the tester will attempt multiple forms of invalid input, including numbers both below and above the range of the options provided, and non-numeric characters. It is expected that

the system will notify the user of the invalid input, reprint the menu, and request new input. The tester will then choose to login as a provider, and repeat the tests for invalid input when asked to input the provider's ID. Invalid input for this field includes anything that does not match a provider ID in the database, the software should handle the invalid input the same way as before, by requesting new input. Finally, the tester will enter a valid provider ID, and it is expected that the software will step into the provider system without error, and this test is complete.

5.2 Provider System Test

The provider system tests will be performed using the same methodology as the smoke tests. However, the tester will make multiple attempts at submitting invalid inputs wherever input is needed. When testing the ability to log a service, the system will request that certain properties of the service be input twice for verification purposes. At this stage the tester will need to repeat the test at least once for each property, and once with multiple properties, giving a non-matching input each time. Each time the system should notify the user which properties do not match, and ask whether the user wants to change each property to the new value, keep the old value, or cancel the service and return to the main provider menu.

To ensure that generating a service report only gathers the data of services provided in the past 7 days by the provider generating the report, a test will be written that compiles a service report when the database has services from multiple providers, including the provider requesting the report, logged over the past 2 weeks. It is expected that the report will contain only those services provided by the provider that requested the report, that were logged in the last 7 days.

The provider directory test is relatively simple as it only involves a lookup into the database. Any errors that can occur here are errors with the database and not with the provider system. As such, simple checking that a valid service can be retrieved and an invalid service fails is sufficient, along with the usual input verification tests for invalid input.

5.3 Manager System Test

Similar to the provider system tests, the manager system tests for report compilation requires minimal testing as the process is non-interactive once the tester starts it. The manager needs to be able to compile a report for all services that each member has received in the last 7 days, and a report for all services that each provider has provided in the last 7 days. A test will be written using the same strategy as the one used for testing the provider generated report.

The last remaining tests for this system are the tests for managing the table of members and providers. The manager needs to be able to add, remove, and update elements of these tables. The strategy for testing this on providers and members is almost identical. A tester will progress to the manager menu, elect first to add, then to update, and finally to remove a member or provider(each test must be run). When adding to either table, the tester will fill in some mock data when prompted, entering incorrectly formatted information where possible(i.e. entering only numbers as a name, etc.), to test the system's ability to handle unwanted input. A helper function will be used to retrieve and display the added profile from the database. It is expected that all data matches what the tester entered. The tester will then choose to update the member or provider that they previously added. The system should request a provider or member ID to retrieve from the database for the change. The tester will provide invalid ID's as part of this test before moving on, it is expected that the operation will terminate and that the tester is returned to the previous menu with a relevant error message. After the tester enters a valid ID, it is expected that the system will provide the tester with a list of fields they can choose to update. Any fields other than the ID number should be part of this list. The tester will attempt to change each field available, testing invalid inputs as part of the process. When the tester has made changes to each field, the old profile will be removed from the database and the new one will be added. This is acceptable because services in the database reference providers and members by using their ID as a key to find their profile. Removing a profile has no impact on service records so long as the profile is replaced quickly. This is expected to take less than a second, and no longer than three. Once again, a helper function will be called to display the profile as it exists in the database. All data should match the new information the tester entered.

Removing a profile is fairly simple to test, which will be done using the profile added earlier in these tests. The tester will need to provide an ID for the profile to remove, upon entering the ID of the profile added previously, the system will display the profile's basic information, a warning that removing the profile will also remove any services associated with that profile, and asks for confirmation from the tester. If the tester cancels, the profile should remain untouched in the database. If the tester confirms the operation, the profile and any associated services should be removed from the database. These cases will all be tested.