**The ChocAn Simulator**

by Team 7, LLC

Portland State University

Science Research and Teaching Center

1719 SW 10th Ave., Suite 247

Portland, Oregon 97201

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Author | Changes |
| 2017-11-11 | 0.1 | Brent McManus | Member/Provider/Service unit tests |
| 2017-11-12 | 0.2 | Edward Koroteev | Template List unit tests |
| 2017-11-12 | 0.3 | Christian Clevinger | CAC unit tests |
| 2017-11-12 | 0.4 | Austin Parish | CAC unit tests |
| 2017-11-12 | 0.5 | Lance Booth | Records unit tests |
| 2017-11-12 | 1.0 | Ray Emory | Introduction, Descriptions, Formatting |
| 2017-11-13 | 1.1 | Ray Emory | Adding to system testing |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc498450684)

[1.1 Purpose and Scope 5](#_Toc498450685)

[1.2 Target Audience 5](#_Toc498450686)

[1.3 Definitions 6](#_Toc498450687)

[1.3.1 Conformance Levels 6](#_Toc498450688)

[1.3.2 Acronyms, Abbreviations, and Initialisms 6](#_Toc498450689)

[1.3.3 Glossary of Terms 7](#_Toc498450690)

[2 Test Plan Description 8](#_Toc498450691)

[2.1 Scope of Testing 8](#_Toc498450692)

[2.2 Testing Schedule 8](#_Toc498450693)

[2.3 Release Criteria 8](#_Toc498450694)

[3 Unit Testing 9](#_Toc498450695)

[3.1 Strategy 9](#_Toc498450696)

[4 Smoke Testing 11](#_Toc498450697)

[4.1 Provider Login 11](#_Toc498450698)

[4.2 Log Service 11](#_Toc498450699)

[4.3 Provider Logout 11](#_Toc498450700)

[4.4 Administrator Login 11](#_Toc498450701)

[4.5 Add Member 11](#_Toc498450702)

[4.6 Run Member Report 11](#_Toc498450703)

[4.7 Add Provider 12](#_Toc498450704)

[4.8 Run Provider Report 12](#_Toc498450705)

[4.9 Administrator Logout 12](#_Toc498450706)

[5 System Testing 13](#_Toc498450707)

[5.1 Provider Login 13](#_Toc498450708)

[5.2 Validate Member Number 13](#_Toc498450709)

[5.3 Log Service 13](#_Toc498450710)

[5.4 Request Services Directory 14](#_Toc498450711)

[5.5 Administrator Login 14](#_Toc498450712)

[5.6 Add Member 14](#_Toc498450713)

[5.7 Modify Member 14](#_Toc498450714)

[5.8 Delete Member 15](#_Toc498450715)

[5.9 Add Provider 15](#_Toc498450716)

[5.10 Modify Provider 15](#_Toc498450717)

[5.11 Delete Provider 16](#_Toc498450718)

[5.12 Run Member Report 16](#_Toc498450719)

[5.13 Run Provider Report 16](#_Toc498450720)

[5.14 Generate Summary Report 16](#_Toc498450721)

[5.15 Generate Weekly Reports 16](#_Toc498450722)

[6 Appendix A – Unit Tests List 17](#_Toc498450723)

[6.1 Member Object Unit Tests 17](#_Toc498450724)

[6.2 Provider Object Unit Tests 17](#_Toc498450725)

[6.3 Service Object Unit Tests 17](#_Toc498450726)

[6.4 Record Object Unit Tests 18](#_Toc498450727)

[6.5 List Template Unit Tests 18](#_Toc498450728)

[6.6 CAC Unit Tests 19](#_Toc498450729)

# Introduction

## Purpose and Scope

This document describes the plan for verification and validation of the Chocoholics Anonymous Management Program System software developed by Team 7. Within this document are a series of unit tests to be implemented during development by Team 7. Expanding on the unit tests are the smoke test and system tests, to guide final validation of the software generated by Team 7.

The test plan outlined in this document builds upon the requirements and design documents generated during previous phases.

## Target Audience

This document is intended for use by the development group working at Team 7, who has been contracted by Chocoholics Anonymous to deliver the data management software.

The test plan outlined within this document is to be followed by the developers at Team 7 in the validation of the data management software for Chocoholics Anonymous.

## Definitions

### Conformance Levels

|  |  |
| --- | --- |
|  |  |
| **Expected** | A keyword used to describe the behavior of the software in the design models assumed by this requirements document. |
| **May** | A keyword that indicates flexibility of choice with no implied preference. |
| **Shall** | A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements. |
| **Should** | A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.” |

### Acronyms, Abbreviations, and Initialisms

|  |  |
| --- | --- |
|  |  |
| **CAC** | Chocoholics Anonymous Control Center. Refers to the primary control interface to be used by actors upon the data management software |
| **ChocAn** | Chocoholics Anonymous |
| **M-PIN** | Member Personal Identification Number |
| **P-PIN** | Provider Personal Identification Number |
|  |  |
|  |  |

### Glossary of Terms

|  |  |
| --- | --- |
|  |  |
| **Administrator (Admin)** | An employee of Chocoholics Anonymous. Has specific permission for managing and controlling the system. |
| **Database** | In the context of this document and software, a database refers to a localized text file that stores information relating to the software. |
| **Member** | A member of Chocoholics Anonymous |
| **Provider** | A healthcare provider for Chocoholics Anonymous. |
| **Service Code** | A 6 digit code corresponding to a service available from a Provider |
| **Service(s)** | Refers to the service provided by a Provider to a Member. |
| **Software** | Unless otherwise noted, refers to the Chocoholics Anonymous Data Processing software. To be created based off of the requirements outlined within this document. |
| **Terminal** | A terminal interface with which to control the software. |
| **User** | A user of the software. Can refer to either an Administrator or a Provider. |
| **Validated** | An M-PIN is recognized as valid by the software. Meaning that the member exists and is current on their payments to ChocAn |

# Test Plan Description

Considerations of the project, as it relates to goals, constraints, and engineering practices.

## Scope of Testing

The testing outlined within this document relates to the functional testing of the ChocAn software. Unit testing shall be carried out for each of the classes outlined in the design document. These unit tests shall validate that each function within each class is capable of carrying out the requirements placed upon them. This shall be accomplished with tests feeding input and validating the output of each function within the classes; both for positive and negative assertions.

The system testing shall outline the validation required for ensuring that the software as a whole is capable of performing to the needs of ChocAn, as described within the requirements documentation.

## Testing Schedule

Unit tests shall be implemented by Team 7 developers alongside program code. This is to ensure that upon completion of the software, functionality of the software shall be near full validation.

As delivery of the database management software to ChocAn is scheduled for November 30th, 2017, system testing shall be carried out in the week leading up to the delivery date. Team 7 will thus be able to create dependable database management software for Chocoholics Anonymous by the expected delivery date.

## Release Criteria

The software shall be deemed acceptable for release to ChocAn upon passing the entirety of the system tests that are outlined within section 5 of this document.

# Unit Testing

See Appendix A. for a list of planned unit test cases.

Unit testing shall cover the individual classes for the ChocAn software: this includes the person class, member class, provider class, services class, administrator class, template list class, records class and the CAC class.

The Member class manages the data for an individual ChocAn Member. Unit tests must validate that the software can identify a member based off of their M-PIN, authenticate their membership status, and create a string to be written to the database.

The Provider class manages the data for an individual ChocAn Provider. Unit tests must validate that the software can identify a provider based off of their P-PIN, authenticate their password, and create a string to be written to the database.

The Service class contains the information for a service that can be offered by Providers. Unit tests must validate that the software can identify a service based off of a given service number, return the fee for a service, and create a string to be written to the database.

The Administrator class contains the information for an administrator. Unit tests must validate that the software can identify a user based off their username, authenticate their password, and create a string to be written to the database.

The Record class contains the record of a service rendered by a ChocAn Provider to a Member. Unit tests must validate that the software can identify if a record contains a specified provider or member based off a given P-Pin or M-Pin, obtain the fee of the service contained within the record, and create a string to be written to the database.

The Template List class contains a collection of member objects, provider objects, service object or administrator objects. It can only contain one type of object, unit tests must be written to ensure that if a list already contains one type of object, a different type of object cannot be added to the list. Unit tests must also be written to validate that the list can identify an object based on a number, and is capable of returning that object to the calling function.

The CAC class is the main interface with a user and contains lists for collections of members, providers, services, administrators and records. As the main interface, the CAC class is responsible for the filtering of user input; therefore unit tests must cover both expected user inputs and unexpected user inputs. The CAC is responsible for not creating any objects with incomplete data.

## Strategy

Unit tests are to be written during code development. This will ensure optimal code coverage. Expected code coverage for each of the aforementioned classes is 85%. As it is not feasible to expect 100% coverage, we expect a target of 85% coverage shall enable Team 7 to deliver a complete and validated software product to ChocAn in a reasonable timeframe.

Omitted from coverage is the file read and file write methods, this shall be covered with system testing; however, the functions that prepare the data for writing shall have unit test coverage.

# Smoke Testing

The smoke test shall test key components of the software through black box testing. Unit tests are to be run prior to the smoke test. See Appendix A for the list of unit test cases.

## Provider Login

A Provider shall be capable of logging into the ChocAn software using their P-PIN and password. Upon authentication, the Provider shall be presented with the provider menu.

## Log Service

A provider shall be able to log a service through the provider menu. They must be asked to enter an M-PIN, service code, date of service rendered and a comment. They must be returned to the provider menu after logging the service.

The database must be updated with the logged service record.

## Provider Logout

A provider shall be able to logout of the software through the provider menu. Upon logout, they must be returned to the login menu.

## Administrator Login

An Administrator shall be capable of logging into the ChocAn software using their username and password. Upon authentication, the Administrator shall be presented with the administrator menu.

## Add Member

An Administrator shall be able to create a new member and add them to the ChocAn database. The Administrator shall be prompted to enter a name, address, city, state, zip code and new member number.

The database must be updated with the new member.

## Run Member Report

An Administrator shall be able to generate a member report. They will be prompted for an M-PIN, the software will then output the member report both to the terminal window and to a file.

## Add Provider

An Administrator shall be able to create a new provider and add them to the ChocAn database. The Administrator shall be prompted to enter a name, address, city, state, zip code and new provider number.

The database must be updated with the new provider.

## Run Provider Report

An Administrator shall be able to generate a provider report. They will be prompted for a P-PIN, the software will then output the provider report both to the terminal window and to a file.

## Administrator Logout

An Administrator shall be able to logout of the software through the administrator menu. Upon logout they must be returned to the login menu.

# System Testing

System testing shall include black box validation of the ChocAn software as a whole. Each path through the software must be validated. Paths that are covered within the smoke test are not included here, it is expected that the smoke test be run prior to commencing system testing.

## Provider Login

A Provider who attempts to login with an invalid P-PIN will be shown messaging that their Provider Number or Password is incorrect.

A Provider who attempts to login with a valid P-PIN but an invalid password will be shown messaging that their Provider Number or Password is incorrect.

## Validate Member Number

A Provider shall be capable of validating the M-PIN of a member. They must be asked to enter the M-PIN, and the system will print the status of that member.

If an M-PIN is entered that does not match an existing member, the user will be messaged that there is no member with that specific M-PIN.

If an M-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an M-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

## Log Service

If when logging a service, an M-PIN that is entered does not match an existing member, the user will be messaged that there is no member with that specific M-PIN.

If when logging a service, an M-PIN that is entered is for a member that is not current, the user will be messaged that they cannot log a service to a suspended member.

If when logging a service, a service code is entered that does not match an existing service, the user will be messaged that there is no service with that specific code.

If when logging a service, an M-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an M-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

If when logging a service, a service code that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of a service code. Invalid structure includes an entry that is not exactly 6 characters or an entry that contains any characters that are not numbers.

## Request Services Directory

A Provider shall be capable of requesting the services directory. This will display the list of services to the terminal window, and then return the Provider to the provider menu.

## Administrator Login

An Administrator who attempts to login with an invalid username will be shown messaging that their username or Password is incorrect.

An Administrator who attempts to login with a valid username but an invalid password will be shown messaging that their username or Password is incorrect.

## Add Member

If when creating a member, an Administrator enters any invalidly structured data, they are messaged that their entry is invalid. Invalid entries for the name include any entry over 25 characters or the empty string. Invalid entries for the address include any entry over 25 characters or the empty string. Invalid entries for the city include any entries over 14 characters or the empty string. Invalid entries for state include any entries that are not an accepted two character state abbreviation. Invalid entries for the zip include any entries that are not exactly five characters or that contain any characters other than numbers.

## Modify Member

An Administrator shall be capable of modifying an existing member’s information. The software will begin by prompting the Administrator to enter an M-PIN for the member. The Administrator shall then be able to modify any of the existing member’s information, including updating their membership status.

If an invalid entry is made for the modification of any member data, the user will be messaged that their entry is invalid. See section 5.6 for what constitutes an invalid entry.

If an M-PIN is entered that does not match an existing member, the user will be messaged that there is no member with that specific M-PIN.

If an M-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an M-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

If any changes are made, the database must be updated with those changes.

## Delete Member

An Administrator shall be capable of removing a Member from the ChocAn database. The software will begin by prompting the Administrator to enter an M-PIN for the member. The Administrator shall then be prompted to confirm the deletion. Upon confirmation the member shall be removed.

If an M-PIN is entered that does not match an existing member, the user will be messaged that there is no member with that specific M-PIN.

If an M-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an M-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

The database must be updated to remove the member.

## Add Provider

If when creating a provider, an Administrator enters any invalidly structured data, they are messaged that their entry is invalid. Invalid entries for the name include any entry over 25 characters or the empty string. Invalid entries for the address include any entry over 25 characters or the empty string. Invalid entries for the city include any entries over 14 characters or the empty string. Invalid entries for state include any entries that are not an accepted two character state abbreviation. Invalid entries for the zip include any entries that are not exactly five characters or that contain any characters other than numbers.

## Modify Provider

An Administrator shall be capable of modifying an existing provider’s information. The software will begin by prompting the Administrator to enter a P-PIN for the provider. The Administrator shall then be able to modify any of the existing provider’s information.

If an invalid entry is made for the modification of any provider data, the user will be messaged that their entry is invalid. See section 5.9 for what constitutes an invalid entry.

If a P-PIN is entered that does not match an existing provider, the user will be messaged that there is no provider with that specific P-PIN.

If a P-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an P-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

If any changes are made, the database must be updated with those changes.

## Delete Provider

An Administrator shall be capable of removing a Provider from the ChocAn database. The software will begin by prompting the Administrator to enter a P-PIN for the provider. The Administrator shall then be prompted to confirm the deletion. Upon confirmation the provider shall be removed.

If a P-PIN is entered that does not match an existing provider, the user will be messaged that there is no provider with that specific P-PIN.

If a P-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an P-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

The database must be updated to remove the provider.

## Run Member Report

If an M-PIN is entered that does not match an existing member, the user will be messaged that there is no member with that specific M-PIN.

If an M-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an M-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

## Run Provider Report

If a P-PIN is entered that does not match an existing provider, the user will be messaged that there is no provider with that specific P-PIN.

If a P-PIN that is not structured properly is entered, the user will be shown messaging indicating the proper formatting of an P-PIN. Invalid structure includes an entry that is not exactly 9 characters or an entry that contains any characters that are not numbers.

## Generate Summary Report

An Administrator shall be capable of generating a Summary Report. The software shall display the summary report to the terminal window and write it to a file.

## Generate Weekly Reports

An Administrator shall be capable of generating the weekly reports. The software shall write to file a member report for all members, a provider report for all providers, and a summary report. These reports will not be displayed in the terminal window.

# Appendix A – Unit Tests List

A list of unit test cases to be implemented for the specified classes. These are just a guideline for currently planned development.

## Member Object Unit Tests

To validate the member class, unit tests must be written to cover the following functions:

* Validate that the prepareWrite function returns a specified string. String is expected to match a value based off of the member object created by the test case.
* Validate that the isActive function returns true if the contained member Boolean is true.
* Validate that the isActive function returns false if the contained member Boolean is false.
* Validate that the equals function returns true if the passed member number matches.
* Validate that the equals function returns false if the passed member number is not a match.

## Provider Object Unit Tests

To validate the provider class, unit tests must be written to cover the following functions:

* Validate that the prepareWrite function returns a specified string. String is expected to match a value based off of the provider object created by the test case.
* Validate that the isPassword function returns true if the passed argument is a match to the contained password.
* Validate that the isPassword function returns false if the passed argument is not a match to the contained password.
* Validate that the equals function returns true if the passed provider number matches.
* Validate that the equals function returns false if the passed provider number is not a match.

## Service Object Unit Tests

To validate the service class, unit tests must be written to cover the following functions:

* Validate that the prepareWrite function returns a specified string. String is expected to match a value based off of the service object created by the test case.
* Validate that the getFee function returns a float that matches the data member in the service object.
* Validate that the equals function returns true if the passed service number matches.
* Validate that the equals function returns false if the passed service number is not a match.

## Record Object Unit Tests

To validate the record class, unit tests must be written to cover the following functions:

* Validate that the prepareWrite function returns a specified string. String is expected to match a value based off of the record object created by the test case.
* Validate that the hasMember function returns true if the passed member number matches the member contained within the record object.
* Validate that the hasMember function returns false if the passed member number is not a match to the contained member object within the record object.
* Validate that the hasProvider function returns true if the passed provider number matches the provider contained within the record object.
* Validate that the hasProvider function returns false if the passed provider number is not a match to the contained provider object within the record object.
* Validate that the getFee function returns the fee of the service object contained within.

## List Template Unit Tests

To validate the list template class, unit tests must be written to cover the following functions:

* Validate that the add function can add a member, provider, service or administrator object to an empty list. Function returns true if the add is successful.
* Validate that the add function can only add a member, provider, service or administrator to a list already containing the same type of object. Coverage will require testing each case. Function to return false.
* Validate that the findNumber function returns true if the passed number matches the number of any object within the list. This shall call the equals function for each object. Test will need to be run for member number, provider number and service numbers.
* Validate that the findNumber function returns false if the passed member number does not the number of any object within the list. This shall call the equals function for each object. Test will need to be run for member number, provider number and service numbers.

## CAC Unit Tests

To validate the CAC class, unit tests must be written to cover the following functions:

* Validate that the loginMenu function can enable a Provider to enter a P-PIN and a password. The function should direct the class object to the provider menu if the combination is accepted.
* Validate that the loginMenu does not authenticate a Provider if an invalid P-PIN is entered.
* Validate that the loginMenu does not authenticate a Provider if an invalid password is entered.
* Validate that the loginMenu function can enable an Administrator to enter a username and a password. The function should direct the class object to the administrator menu if the combination is accepted.
* Validate that the loginMenu does not authenticate an Administrator if an invalid username is entered.
* Validate that the loginMenu does not authenticate an Administrator if an invalid password is entered.
* Validate that the validateMember function will only accept a string that is 9 characters long and contains only digits.
* Validate that the validateMember function can pass the entered M-PIN to the findNumber function of the Member List, and that the function returns the return value of the findNumber function.
* Validate that the logService function will only accept an M-PIN string that is 9 characters long and contains only digits.
* Validate that the logService function will only accept a service code string that is 6 characters long and contains only digits.
* Validate that the logService function will only accept a date string that adheres to the format YYYY-MM-DD. Where YYYY must be a year equal or less than the current year, MM must be a valid month (01-12) and DD must be a valid day (01-31).
* Validate that the logService function will only accept a comment string that is fewer than 100 characters.
* Validate that the logService function can create a record object containing member and service objects that were retrieved from their respective lists.
* Validate that the logout function can log a Provider out of the provider menu, returning the class object to the login menu.
* Validate that the addMember function can enable an Administrator to create a new member object.
* Validate that the addMember function will only create a new member object if the entered name string is not empty and has 25 or fewer characters.
* Validate that the addMember function will only create a new member object if the entered street address string is not empty and has 25 or fewer characters.
* Validate that the addMember function will only create a new member object if the entered city string is not empty and has 14 or fewer characters.
* Validate that the addMember function will only create a new member object if the entered state string is an existing state abbreviation.
* Validate that the addMember function will only create a new member object if the entered zip string is exactly five characters and contains only digits.
* Validate that the addMember function will only create a new member object if the entered M-PIN is exactly nine characters, contains only digits and does not belong to an existing member within the member list.
* Validate that the addProvider function can enable an Administrator to create a new provider object.
* Validate that the addProvider function will only create a new provider object if the entered name string is not empty and has 25 or fewer characters.
* Validate that the addProvider function will only create a new provider object if the entered street address string is not empty and has 25 or fewer characters.
* Validate that the addProvider function will only create a new provider object if the entered city string is not empty and has 14 or fewer characters.
* Validate that the addProvider function will only create a new provider object if the entered state string is an existing state abbreviation.
* Validate that the addProvider function will only create a new provider object if the entered zip string is exactly five characters and contains only digits.
* Validate that the addProvider function will only create a new provider object if the entered M-PIN is exactly nine characters, contains only digits and does not belong to an existing provider within the provider list.
* Validate that the modifyMember function can retrieve a member object from the member list based off an M-PIN string.
* Validate that the modifyMember function only modifies the member object if the entered name string is not empty and has 25 or fewer characters.
* Validate that the modifyMember function only modifies the member object if the entered address string is not empty and has 25 or fewer characters.
* Validate that the modifyMember function only modifies the member object if the entered city string is not empty and has 14 or fewer characters.
* Validate that the modifyMember function only modifies the member object if the entered state string is an existing state abbreviation.
* Validate that the modifyMember function only modifies the member object if the entered zip string is exactly five characters and contains only digits.
* Validate that the modifyMember function can modify the validation status of the member object.
* Validate that the modifyProvider function can retrieve a provider object from the provider list based off a P-PIN string.
* Validate that the modifyProvider function only modifies the provider object if the entered name string is not empty and has 25 or fewer characters.
* Validate that the modifyProvider function only modifies the provider object if the entered address string is not empty and has 25 or fewer characters.
* Validate that the modifyProvider function only modifies the provider object if the entered city string is not empty and has 14 or fewer characters.
* Validate that the modifyProvider function only modifies the provider object if the entered state string is an existing state abbreviation.
* Validate that the modifyProvider function only modifies the provider object if the entered zip string is exactly five characters and contains only digits.
* Validate that the logout function can log an Administrator out of the administrator menu, returning the class object to the login menu.