Group #4

Christopher Juncker

Justin Greever

Samantha Zeigler

Tori Anderson

Naya Mairena

Ian Guy

Dan Jang

Term Project: *ChocAn*

Test Plan Document

Table of Contents

1 Introduction

This document contains information on and describes the process of testing the ChocAn data processing software being developed. Because this software is going to be used in multiple ways by various groups of people, the test plan will have to take into account a notable amount of possible use cases, as well as how certain use cases need to be separated from each other.

The test plan description section will further detail the specific requirements of the testing, and the unit testing section will discuss specific strategies which will be used during testing.

Following this will be smoke testing, and finally system testing - each with individual test cases. System testing cases will be split into user interface tests and report generation tests. Lastly, also under system testing, there will be tests for manager user interfaces and menu integrations.

1.1 Purpose and Scope

The purpose of this document is to provide insight into the testing process of the ChocAn data processing software.

The test plan description should further elaborate on the scope of testing certain items. As for the scope of individual testing scenarios, they should describe the strategy behind the test, what issues the test uncovered, and what solutions were put in place (if needed) - and how these solutions now pass the previously failed test cases.

Beyond this, a debrief section should gather all changes made during the testing process to summarize the effect this process had on the development of the ChocAn data processing software.

1.2 Target Audience

The target audience of this document is the stakeholders of the ChocAn system. This includes the management team, care providers, and individuals who will be servicing the software long-term.

While not directly involved in the development of this software, it will help these stakeholders to understand the improvement process which has been undertaken during the development of their software.

Having this understanding will better allow for the stakeholders to suggest additions or improvements further on in the development process, as well as for them to better take into consideration the desires of users (again, later in the development process - most likely after deployment).

While this stage is less about improvement and more focused on verification of what has been developed so far, it’s useful for allowing a review of what kind of work is being done and the quality of said work.

1.3 Terms and Definitions

Chocoholics Anonymous (ChocAn)	An organization dedicated to helping people addicted to chocolate in all its glorious forms.
ChocAn Data Center (CDC)	The data center which contains member and provider information.
Member	A person who pays a monthly fee to ChocAn and is entitled to unlimited consultations and treatments with providers.
Provider	A health care professional who provides services to members.
Terminal	A computer terminal operated by a provider which scans member cards and which serves as an interface to communicate with the Software Product.
Manager Terminal	A terminal with elevated privileges that can request individual reports at any time.
Summary Report	A weekly report is provided to the manager of accounts payable listing all providers that need to be paid.
Mock Object	A simulated object that mimics the behavior of the real object in a controlled environment.

2 Test Plan Description

The *Chocoholics Anonymous* software system test plan consists of the following stages: Unit Testing, Smoke Testing, and System Testing. These phases will help test and help identify defects throughout each phase and process of the test plan.

Specifically, the test plan would showcase the carrying out of the specific types of testing activities that are paramount to the whole-and-complete test-plan schedule. Furthermore, the test plan is useful as an instrumental guide to testing software-suite goals, the scope of testing for the project, as well as particular testing events that are scheduled.

2.1 Scope of Testing

Scope of testing will cover and describe the areas of the *Chocoholics Anonymous* software that require testing; those features and areas that require testing will be considered in scope and those that do not require testing at this time, are considered out-of-scope.

In-Scope: The *Chocoholics Anonymous* software will be tested on different operating systems & environments - as well as with different users.

In-Scope:

- Authentication System & Interface
  - Logging in
  - Logging out
- Patient System & Interface
  - Messaging Provider(s)
  - Accessing Record(s)
  - Scheduling Appointment(s)

- Healthcare Provider System & Interface
    - Messaging Patient(s)
    - Accessing Record(s)
    - Managing Schedule(s) & Appointment(s)
- Payment System & Interface
    - Viewing Bills as Patient(s)
    - Paying Bill(s)
    - As Provider, Sending Bill(s)
- Database Functionality & System
    - Data Creation Check
    - Data Management Check
    - Data Security Check

Out-of-Scope:

- Non-Python compatible environment testing
- Dedicated mobile application testing
- Mobile platform testing

2.2 Testing Schedule

1. **Authentication System & Interface**

   Start Date: November 31st, 2021

   End Date: December 3rd, 2021

   Testing Steps

    - Input login credentials into the login form.
    - Submit form for authentication.

   Tasks

    - Providing valid login credentials
    - Providing invalid login credentials

Responsibilities

- Ensuring valid credentials are allowed access to correct functions.
- Displaying an invalid login message for invalid credentials.

Reviewing Guide

- After entering a specific set of credentials for a test case, a hypothetical provider verifies that the specific set of access for the test-provider account is granted.
- Entering both small and large length test credentials to verify against buffer overflow attacks & other input-based authentication hijackings.

2. **Provider System & Interface**

Start Date: December 1st, 2021

End Date: December 3rd, 2021

Testing Steps

- Select valid menu items.
  - Select valid sub-menu
- Navigate back to the main menu.
- Create a new patient.
  - Delete patient
- Schedule a patient for a visit.
- Create a note for a specific patient.
- Create date of service provided.
  - List services provided
- Change patient information

Tasks

- Successfully create patient filling needed information on patient
  - Change patient information and check it saved
- Schedule created patient for the visit
- Log visit as completed

    - Input date of service
    - Input services for the visit
- Create a note for the patient
- Delete patient

Responsibilities

- Ensure patient records are only accessible by authorized providers and to the specific patient.
- Display patient data correctly and legibly with correct formatting.

Reviewing Guide

- After adding example patient records with specific providers access to said records, login as each specific provider to verify access to the example patient records & verify non-access to unauthorized patient records.
- After adding large-data format patient records, verify correct displaying and formatting.

3. **Report Generation Systems**

Start Date: December 2nd, 2021

End Date: December 3rd, 2021

Testing Steps

- Select valid menu items.
    - Select valid sub-menu
- Navigate back to the main menu.
- Generate member summary reports.
    - Specific patients selected in the database
- Generate provider summary reports.
    - Specific provider’s information in the database
    - Display provider’s services
    - Display provider’s patients and their information

    - Display Fees needing to be paid of said patient
    - Display total number of services
    - Display total fee of that week
- Generate manager summary reports.
    - Lists all providers in the database
        - Display each service of each provider
    - Display total number of consultations
    - Display total fee of all consultations

Tasks

- Have the number of providers in the database.
- Have the number of consultations from the database.
- Have the overall fees & transactions across all providers.
- Be able to generate Member Services Reports.
- Be able to generate Provider Services Reports.
- Be able to generate Electronic Funds Transfer (EFT) Reports.
- Be able to generate Summary Reports.

Responsibilities

- Ensure data found in generated reports are accurate - as well as legible and formatted correctly.
- Interweaved management extended report functionalities with proper authentication.

Reviewing Guide

- Enter test-case patient data & records, then generate a full report with a specific set of provider credentials. Verify only authorized patient data & records, that is accessible & available to the specific set of provider credentials, are included in the generated report. Ensure no access violations have been incurred.

- Log in as a test-case management member with elevated privileges. Generate reports of management-level or lower patient data & records. Verify accurate data and records with correct formatting.

4. **Database Functionality & System**

   Start Date: December 3rd, 2021

   End Date: December 3rd, 2021

   Testing Steps

   - Perform a test to check for an existing database.
     - Create a database if none is found.
   - Check if Tables exist in the database.
     - Create Tables if none are found.
   - Check if Tables pass read/write tests.
     - Add a new test entry.
     - Search and read test entries.
   - Search for existing data in the database.
     - Check proper data is returned.
     - Update existing data and send it back to the database.
     - Search for altered data and ensure updates are there.
     - Search for altered data again and issue the delete command.
     - Search for altered/deleted data and pass if not found.
   - Test/Validate data passed in from frontend/user functions
     - Pass invalid data but with too many arguments.
     - Ensure the system truncates extra arguments that don’t pass validation or return an error.
     - Pass in valid data with the proper amount of arguments.
     - Pass in invalid data to ensure an error is returned.
     - Return NULL/Not found message to user.

    - Return proper data from the database that was requested in the search query.

Responsibilities

- Ensure the Database is set up and ready for the software to store information in.
- Ensure that data can be added/edited/deleted from the database
- Ensure that the menu/user interface functions work properly with the database management system.

Reviewing Guide

- When a system is initially installed at a facility to ensure the Database is available for storing the appropriate datasets.
- When a new system function is created, the function must be validated to work with the Database system properly.

2.3 Release Criteria

Release testing is the process of having people outside of the development team test the system at a specific release. Release testing is implemented to convince the supplier that the system is usable. The purpose of this is to check that the system meets its requirements and is ready for external use, it is not testing for bugs. This is why it is better to have a team that has not worked on the development of the system do the release testing.

The criteria that our system must meet before being deployed are having a functional terminal interface, functional database, and a functional report generator. These are the main components of functionality that the user will be utilizing when accessing our software.

3 Unit Testing

Testing individual components that are completely isolated from the rest of the software is the purpose of unit testing. Unit testing will allow for a quick and easy search for bugs in the software. Individual components in the software are things such as functions, classes, or the interface.

The purpose of unit testing is to focus on the functionality of the software by testing the individual components to their full capabilities. It is important for the software to be correct so that the entire system functions properly.

This section will describe the unit tests that will be performed on the system. We must describe the functionality of each component, what tests will be performed on them, and the outcomes that are to be expected. Finding errors while testing must be recorded, fixed, and then tested again. The unit tests must test a range of all possible scenarios so that errors are less likely to occur.

The specific unit tests to be tested in our software will be the terminal interface, the database, bill report generation, and the menu options/functionality. Testing the terminal interface is the most important part of the software because it will emulate the functionality of the terminal that the providers and managers will be utilizing on a daily basis.

We need to make sure that the input and output are managed properly. The functionality for the database will keep track of all the data that is involved with the ChocAn memberships. This is also critical to test because the data is the entire purpose of the software we created.

Generating the bill reports will utilize the database to gather the correct information for that specific report. Testing this part allows us to find errors when generating reports because the collected data may be sensitive and using the incorrect data of a specified member could result in issues (especially security issues).

3.1 Strategy

All events that cause change to the state of the object should be simulated in the unit testing process. It is important to pick a strategy of unit test cases that are effective.

This means that when the expected test cases are used, the components being tested do what they should and that if there are any defects, they are to be revealed by the test cases.

A unit testing strategy that can be helpful is using mock objects. Working with a database that is not yet implemented can be difficult for testing parts of our code that depend on the database, so using mock objects allows for testing of these parts (such as the menu selection).

Using mock objects will also need to be used for testing the part of our code that generates the weekly file of services provided. This main accounting procedure is done every Friday at midnight but to be able to test this portion of our code, we need to create a mock object that will simulate this timed event. Mock testing will be helpful before doing system testing where all parts of the system will be integrated.

When it comes to testing the main interface of our system, we need to focus on the specific input of the data. The input that the providers enter is important data about each ChocAn member they interact with.

We must have our software double-check each piece of data before it is sent off to the database. A way to do this is to have conditional statements with specific data ranges that are allowed and not allowed. When the data entered is out of bounds, an error must be displayed and the provider must enter the correct data. The data will not be finalized until the correct data is entered.

For this to function properly, we must create unit testing of each piece of code that takes in data. For unit testing, we will specify ranges, mock data, and data combinations to properly test our interface code.

For integrated testing, we will make sure that the interface functions properly with the data from the database.

For testing the database, the focus must be the data organization. The organization of the data within the database is critical for allowing proper data entry and retrieval. If necessary, there should also be a section in the database that contains private data that can only be accessed by ChocAn management. Methods for testing the database will include pulling data from exact locations requested, ensuring that data is stored in the correct manner, and understanding the categorization of the data by rows and columns. Unit testing of the database will mainly focus on the proper functionality of holding and releasing mock data. Integrated testing of the database will consist of retrieving the correct data requested, adding data to the correct location, and displaying the data as requested from the menu interface.

The report generator for our system must also be tested for accuracy. The weekly report generation should neatly display the data requested. For the unit testing of this portion, we will only focus on the formatting of the output with mock data. The integration testing will make sure to pull the correct data from the database and do the proper calculations with the specified data and then will display the results. Calculations will be required for generating the total amounts for billing of the data collected per time frame. For our system, we must create weekly billing reports with the total amounts to be charged for that week. Making sure that the calculations function properly is extremely important because it will avoid mistakes with finances.

4 Smoke Testing

The purpose of smoke testing is to ensure that the basic features of the software appear to function correctly. This section of basic tests will be performed before the System Tests. Only when it has been confirmed that the software is able to operate on a basic level will the full suite of System Tests be initialized. (Until the Smoke Tests are able to successfully complete, there is no guarantee that the System Tests will be able to run at all.) Our Smoke Testing routine will run the following series of tests in order to determine that the basic functionality of the program is intact:

4.1 Run Test

The purpose of this initial test is to determine whether or not the Data Processing Software is able to run at all. This test involves initializing the program startup routine. The test passes if the program is able to start up successfully. The Run Test will perform the following actions:

1. Send a correct command to the operating system signaling it to initialize an instance of the Data Processing program.

The requirements for this test to pass are as follows:

1. The software must not generate any compile-time errors.
2. The software must not generate any run-time errors.
3. The software must produce the expected terminal output which represents that it has initialized itself correctly. (This output is likely to be the initial user login menu, or whatever text is first output to the terminal as outlined in the design document.)

4.2 Login Test

The purpose of this test is to ensure that it is possible to log in to the software system. This test involves attempting to gain access to the system with a valid set of credentials. This test passes if the program is able to log in correctly and access the first level of protected user menus. The Login Test will perform the following actions:

1. Fill out the login form with a valid Provider Identity Number.
2. Submit the login form.

The requirements for this test to pass are as follows:

1. The software must not generate any run-time errors.
2. The software must produce the expected terminal output which represents that it has accepted the login information, and has granted access to the main user menu system.

4.3 User Interface / Menu Test

The purpose of this test is to confirm that the user menus are functional and that they are able to be displayed correctly. The test software will attempt to visit all menus in order to determine that the menu functionality is intact. This test will not determine whether the menu actions have been completed successfully. The goal is only to determine that the menu system is correct and navigable as intended. The User Interface Test will perform the following actions:

1. Sequentially select each valid menu item. For each menu item,
    a. Sequentially select any valid sub-menu items.
    b. Attempt to navigate back to the main menu.

The requirements for this test to pass are as follows:

1. The software must not generate any run-time errors.

2. The software must produce the expected terminal output for each menu item that is selected.
3. The software must respond correctly to all menu navigation attempts.

5 System Testing

System testing is when most or all of the components in the system are integrated and the entire system is tested as a whole. It is important to focus on testing the interactions between each component because they must be compatible, interact properly, and share/transfer the right data. System testing is a collective process and since we are working in a team, it is important to make sure all our individual components work together. Some errors may not be visible until all the components are put together.

The main focus on the test cases should focus on the integration of the interface and the database. These are the main two components that are being implemented for our data processing software. It is crucial that the correct data is being stored and shared for the proper members of ChocAn. Another component that is important to emulate for our third-party software is the report generator.

5.1 User Interface Test

The goal of the User Interface Test is to test the interface as a provider. The functionality that the provider is allowed consists of entering information of a new member or existing member, requesting services, etc. This will be testing the integration of the interface and the database. The data entered by the provider should be correct and stored in the correct database. The data requested by the provider should be the correct data pulled by the database.

5.1.1 New Member Subtest

The New Member subtest will test the user creation process by using the user interface to create a new member. The New Member test tries to create two new members:

1. Invalid Member: An attempt is made to create a user with invalid information. This test passes if the user interface displays the correct error message identifying the invalid information.
2. Valid Member: An attempt is made to create a user by providing valid user information. This test passes if the user interface displays the correct success message stating that the user has been added to the user directory.

5.1.2 Member Lookup Subtest

The Member Lookup subtest will test the member lookup function of the member directory by using The user interface to search for members. The Member Lookup subtest tries to look up two types of patients:

1. Non-existing Patient: An attempt is made to look up a patient who does not exist in the member directory. This test passes if the user interface displays the correct error message stating that the patient was not found.
2. Existing Patient: An attempt is made to look up a patient who is known to exist in the member directory. This test passes if the user interface displays the correct patient information for the user.

5.1.3 Provider Lookup Subtest

The Provider Lookup subtest will test the provider lookup function of the provider directory by using the user interface to search for providers. The Provider Lookup subtest tries to look up two types of providers:

1. Non-existing Provider: An attempt is made to look up a user who does not exist in the provider directory. This test passes if the user interface displays the correct error message stating that the user was not found.
2. Existing Provider: An attempt is made to look up a user who is known to exist in the provider directory. This test passes if the user interface displays the correct provider information for the user.

5.1.4 Service Lookup Subtest

The Service Lookup subtest will test the service lookup function of the service directory by using the user interface to lookup service codes. The Service Lookup subtest tries to look up two types of service:

1. Non-existing Service: An attempt is made to look up a service code that does not exist in the service directory. This test passes if the user interface displays the correct error message stating that the service code was not found.
2. Existing Service: An attempt is made to look up a service code that is known to exist in the service directory. This test passes if the user interface displays the correct service information for the service code which was provided.

5.2 Report Generation Test

The purpose of the Report Generation test suite is to test the report generation for third-party software. The report generation is requested weekly and contains specific information. This will be testing the integration of the database with the code that will generate a report and save it to disk.

5.2.1 Member Report Subtest

The Member Report subtest will test the report generation abilities of the software by generating a member report. After issuing a request for the report to be generated, the subtest will then inspect the report to determine its validity. This test passes if the following conditions are met:

1. The report file exists.
2. The report file is not empty.
3. The report file is formatted appropriately.

5.2.2 Provider Report Subtest

The Provider Report subtest will test the report generation abilities of the software by generating a provider report. After issuing a request for the report to be generated, the subtest will then inspect the report to determine its validity. This test passes if the following conditions are met:

1. The report file exists.
2. The report file is not empty.
3. The report file is formatted appropriately.

5.2.3 Electronic Funds Transfer (EFT) Report Subtest

The Electronic Funds Transfer Report subtest will test the report generation abilities of the software by generating an EFT report. After issuing a request for the report to be generated, the subtest will then inspect the report to determine its validity. This test passes if the following conditions are met:

1. The report file exists.
2. The report file is not empty.
3. The report file is formatted appropriately.

5.2.4 Summary Report Subtest

The Summary Report subtest will test the report generation abilities of the software by generating a summary report. After issuing a request for the report to be generated, the subtest will then inspect the report to determine its validity. This test passes if the following conditions are met:

1. The report file exists.
2. The report file is not empty.
3. The report file is formatted appropriately.

5.3 Manager Interface Test

The goal of the Manager Interface Test is to test the interface as a manager trying to access specific information from the database that may be protected. This specific data may be sensitive, thus should be safely stored and not allow access to unauthorized users.

The Manager Interface test will confirm that the privileged manager-level functions are only accessible by authorized users who have been granted the appropriate permissions. The Manager Interface Test will perform two separate checks:

1. Unauthorized User: The test program will log in to the system using the credentials of a valid user who has not been granted manager-level permissions. The program will then attempt to run manager-level functions. This test passes if the attempted operations fail or are otherwise shown to be inaccessible to the unauthorized user.
2. Authorized User: The test program will log in to the system using the credentials of a valid user who has been granted manager-level permissions. The program will then attempt to run manager-level functions. This test passes if the attempted operations are available to the program and if the operations succeed.

5.4 Menu Integration Test

The purpose of the Menu Integration Test is to test the menu interface with the integration of the database and weekly report generation. This will make sure that every choice in the main menu does what it should and allows the user to have an easy menu interface that flows with what they request. The menu interface should connect to the database for accessing data. The menu interface should connect with the report generator that will generate and display a weekly report with the correct data requested in the report generation component. The Menu Integration Test will be comprised of the following subtests:

5.4.1 Menu Selection Subtest

The Menu Selection subtest will test each valid menu selection. The subtest will also test several invalid menu options. The subtest passes if each valid menu selection results in the correct menu being displayed to the user and if each invalid menu selection results in an appropriate error message confirming that the invalid input has been handled correctly.

5.4.2 Form Manager Subtest

The Form Manager subtest will test each form that is managed by the software. These forms will connect with the database. This subtest will fill each form out with both valid and invalid data. The subtest passes only if each form correctly accepts the valid data and displays an appropriate response. Furthermore, the subtest passes only if each form correctly recognizes the invalid data and displays an error message appropriately.

5.5 Database Functionality Tests

The purpose of the Database Functionality Tests is to ensure that the proper database structure has been created, that the helper functions are able to read/retrieve/edit/delete data within the database, and that any extra values passed to the functions are either truncated or return an error message.

5.5.1 Database Creation Subtest

The Database Creation Subtest will test if the database currently exists if the database has data stored already, and if no database exists, the creation of a new database with required tables added. The tests will only pass if a database exists with the proper tables (with or without existing data in the tables), or if no database exists and the software can create/connect to a new database and ensure proper tables and structures are created.

5.5.2 Database Read/Write Subtest

The Database Read/Write Subtest will test the ability to read data that is stored in the database and pass only if valid data is found. The written test will create a new test entry, and verify that the data exists in the database during the reading subtest. It will fail if the data cannot be found, and will pass if the data is present in the database that matches the query. This test relies on the Database Creation Subtest to return a pass, otherwise, it will fail if no database is found, or the tables are not present in the database.

5.5.3 Database Update/Delete Subtest

The Database Update/Delete Subtest will test the ability to retrieve a value from the database, alter the data, write it back to the database, then attempt to locate and read the newly edited data. It will pass if the new data is found, and fail if it cannot find the data within the database. The delete subtest will then search for the data that was just edited and verified to be in the database, and remove/drop the information from the database, then perform a search for the data within the database. It will pass if the data is not found, and will fail if the data is found.

5.5.4 Database Helper Function Subtest

The Database Helper Function Subtest will test the database helper functions' ability to take arguments passed from the frontend/menu interface and match the values to valid entries in the database. The test will pass if all passed values match a valid entry in the database. If the arguments are incorrect, or if there are too many arguments, the test will fail. These tests will cover all tables in the database and their corresponding functions. If no database is found, or no matching tables found, the test will fail.