Calico Randall, Anna Kawasaki, Viktoriya Petrova,
Andy Truong, Jonathan Nguyen, Natasha Naylor

The ChocAn Simulator

Design Document

Table of Contents

1 Introduction

This document will provide a detailed overview of the design and overall architecture of the Chocoholics Anonymous (ChocAn) data processing software system, including any constraints and dependencies that must be considered.

1.1 Purpose and Scope

The purpose of this design document is to provide a description of the design of the system fully and accurately to allow for software development to proceed with an understanding of what the requirements for the system are, and how they will be implemented. This document will also include an overview of the system architecture, as well as any supporting Unified Modeling Language (UML) diagrams that will help the development team visualize the overall system.
The scope and focus of this document is to lay out a detailed description of what the system features will be, and how they will be implemented. In addition, the aim of this document is to provide the information required to have developers start constructing the system software according to the ChocAn organization's specifications previously outlined in the requirements document. This document will not include a plan to test the system, as this will come in a separate document at a later stage of development.

1.2 Target Audience

The target audience for this document are the developers of the project, including the project manager, project team members, and software development team members. Developers should use this document to understand specifics of how the system will work, how each requirement should be implemented, and the structure of the overall architecture. Other stakeholders may also use this document to verify that the design is accurate to their desires and requirements for the system. End users, such as ChocAn managers, can review the document and provide feedback on how the effectiveness and usability of the features looks on their end. Any requests to change this document will be discussed among stakeholders and incorporated into this document before implementation may begin. As ChocAn managers have a higher level of access to the system than other stakeholders such as providers and members, they may also use this document as a resource to visualize how the system is structured to gain a better understanding of its functionality as it's applicable to their job duties.

1.3 Terms and Definitions

1. Acme - Acme Accounting Services
2. ChocAn - Chocoholics Anonymous
3. DOS - Date of service
4. EFT - Electronic Funds Transfer
5. IDE - Integrated Development Environment
6. ID number - Identification number
7. Manager - Administrator for the ChocAn organization
8. Manager Terminal - Terminal to be used by ChocAn managers to request data records and to add, delete, and update member and provider records
9. Member/Customer - Individuals enrolled in ChocAn memberships
10. Member/Membership Fee - Monthly fee billed to each registered member
11. Provider - Healthcare professional in the ChocAn network
12. Provider Directory - A directory to be used by the provider to locate the appropriate procedure codes for the services provided
13. Provider Terminal - Terminal to be used by providers at the time of service to verify active ChocAn memberships and bill for ChocAn services
14. Service Fee - Commission that is to be paid to the provider
15. UML - Unified Modeling Language

2 Design Considerations

In this section, we will describe the specifications of the system design and address areas that may affect the requirements, design, or operational concepts of the system. Constraints and dependencies of the system will address the non-functional requirements and collaborations with third-party organizations previously outlined in the requirements document. This section will also address our chosen software process model, including its advantages over other considered models.

2.1 Constraints and Dependencies

This section will address the functional and non-functional requirements previously determined in the requirements document, including any system constraints and dependencies.

2.1.1 System Security

The system will have different levels of access for different users to ensure a high level of system security. This will be achieved through an object oriented design which has the ability to create levels of access control for the different system users. This design implies that references to objects will be used in order to prevent users from accessing restricted information and prevent malicious code from accessing clinic records. This will allow for managers to have a higher level of functional and data access while providers/members to have a much more limited scope.

2.1.2 Performance and Reliability

The system will perform at a consistent speed and will be reliable. Since performance and reliability are dynamic aspects of the software, they cannot be fully met during implementation. Creating a reliable and fast program requires frequent testing and feedback from the users. Initial benchmarks that the software will strive to meet include never exceeding three seconds of response time when inputting, accessing, or verifying data, having less than 3% of failures will result in data corruption, and less than 1% of failures will result in system shut-down. Additionally, during any downtime, member and provider information can still be accessed through external files that are updated consistently with system use.

2.1.3 Usability and Testability

To improve usability, the software will display on the terminal any user error encountered (such as invalid input), the specifics of the error, and an example of a proper use case. Additionally, before exiting the system, users shall have the option to provide anonymous feedback on their experience with the software. This feedback will be stored in an external file so that those maintaining the software can view and address any significant issues. To improve testability, the software will avoid using non-deterministic methods, it will include inversion of control by separating decision making code from action code, and will strive towards a highly modular design. This will allow for unit testing, smoke testing, and system testing to be done during the testing portion of the Waterfall model.

2.1.4 Language

All software implementation is restricted by the programming language chosen. The language chosen for this software is C++ due to its ability to support object oriented programming, advantages in scalability, and its feature of portability.

2.1.5 Environment

All software implementation is restricted by the programming environment that it is written in. The environment chosen for this software is Visual Studio Code with a C/C++ extension. The version control system chosen is GitHub, which can be linked directly to Visual Studio Code. The software will also have the ability to be run and tested in a command line linux environment as well as other available IDEs that have support for C++.

2.1.6 Acme Accounting Services

The ChocAn's Data Center membership records are dependent on our collaboration with Acme Accounting Services in order to update records every evening at 9:00PM PST. Acme is a third-party organization who is responsible for recording payment of membership fees, suspending members whose fees are overdue, and reinstating suspended members who have paid their dues. The software required by Acme to complete such financial procedures is to be designed and created by a third-party organization. Our system will be responsible for storing all membership records such as member demographics, but excluding member's method of payments, in the external file storage system.

2.1.7 Banking System

The ChocAn data processing software is dependent on our collaboration with a third-party organization that is responsible for the electronic funds transfer (EFT) banking computer system, which credits the appropriate amount owed to all ChocAn providers. Our software will only have the ability to store all relevant EFT data into the external file storage system, in order to create a weekly report to be sent to our aforementioned third-party collaborator.

2.1.8 Communications System

The Data Center's summary reports are dependent on the third party communications software which is responsible for sending out the emails containing the weekly reports. An email will be sent to each member who consulted a ChocAn provider containing a full list of services rendered for that week. An email will be sent to each provider who has billed ChocAn containing a full list of services rendered to ChocAn members. By request of a provider, an email will be sent containing a full list of ChocAn service names and corresponding service codes and fees. The data center software will only have the ability to store the summary reports data in an external file storage system and display the reports to the appropriate terminal.

2.1.9 Terminal Design

The system is dependent on a third-party organization to design and build the hardware of the provider and manager terminals. The data center software is only responsible for implementing the provider and manager's terminal functionality including reading and writing data to the external file storage system, validating users, and displaying summary reports.

2.2 Methodology: The Waterfall Model

Since the development of this system relies on a plan-driven process, we have decided to follow the waterfall software development model. With this model, it requires us to have unanimous approval on a finished state of the software development process from system stakeholders such as the ChocAn organization’s administrative team and managers, and the software development team before continuing on to the next phase of development. This model was a necessary choice, as our software is reliant and therefore limited based on both the provider and manager terminal hardware design and construction. With this method in place, it is essential for active collaboration and responsiveness between the software development team and the ChocAn organization in order to adjust the requirement expectations and design in a timely manner, so that our development team can move on to implementation and be completed on time. The waterfall model can be broken down into five stages of development.

2.2.1 Requirements Analysis and Definition

The establishment of the ChocAn’s data processing system’s intended services, constraints, and goals through collaboration of system stakeholders. This stage of the development process was completed and approved by stakeholders on January 31, 2022.

2.2.2 System Software Design

Our current stage of development; during this stage, this design process allocates the requirements for the software design. We establish the overall system architecture and identify our fundamental software system abstractions and their relationships with one another. This stage is set to be completed upon approval by system stakeholders by February 14, 2022.

2.2.3 Implementation and Unit Testing

With the system requirements and design clearly defined, the software development team begins to realize the software design into a set of program units. The process of unit testing demands verification that each program unit meets its requirements. This process must be completed and approved before the planned start date of integration and system testing on February 28, 2022.

2.2.4 Integration and System Testing

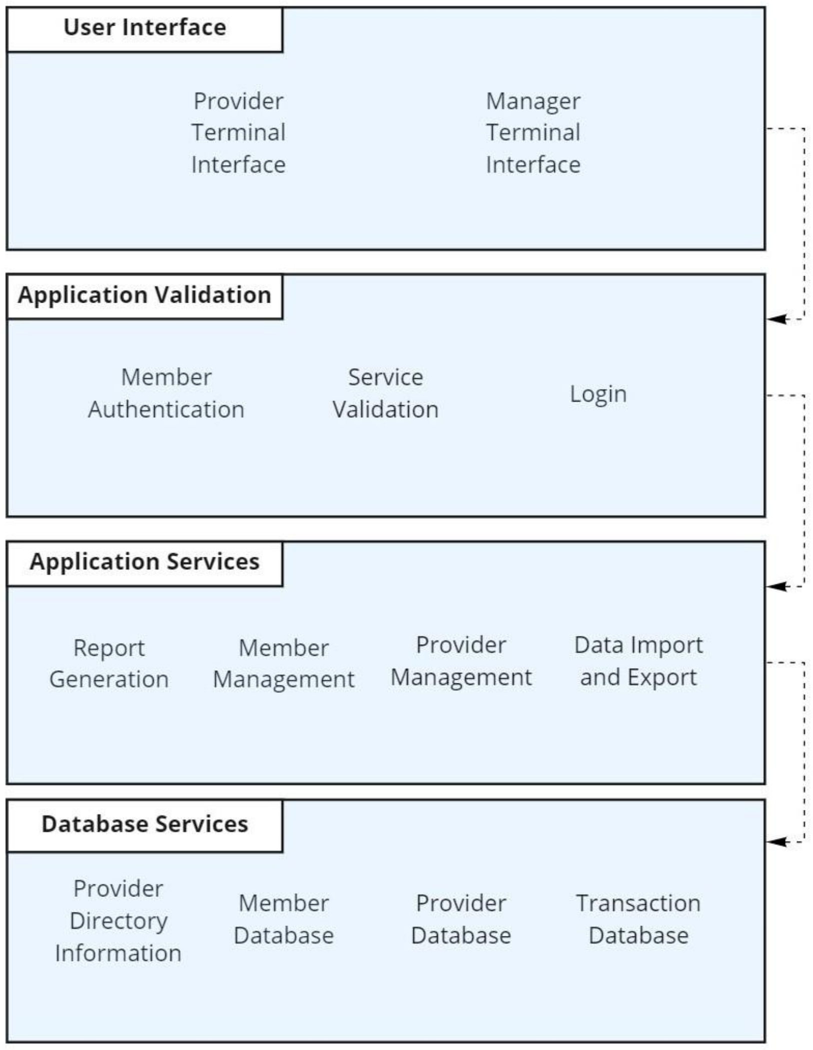
The software development team integrates the individual program units completed during the previous stage of development and begins to test the ChocAn system as a complete unit to ensure that the software requirements have been satisfied. A formal test plan will be completed upon approval by no later than February 28, 2022. Once this stage is completed and approved, the software system will be delivered to the ChocAn organization.

2.2.5 Operation and Maintenance

The ChocAn data processing software system is expected to be completed, approved, and delivered to the ChocAn organization by no later than March 11, 2022. Once the software has been delivered, the system will be installed and providers and managers will be able to use it. It has been previously determined that any maintenance or evolution of the software is the responsibility of the ChocAn organization and/or whomever they decide to handle said maintenance.

3 System Overview

The ChocAn data processing software will follow a layered architectural pattern, in order to achieve proper development separation and independence that allows localized changes. For this system, functionality is organized into four separate layers: User Interface, Application Validation, Application Services, and Database Services. Each layer is only dependent on the services and facilities provided by the layer directly beneath it. We have chosen this approach because it supports incremental development, and also has the benefit of being both portable and changeable. An overview of this architecture is visualized below:



3.1 User Interface

The User Interface layer consists of the provider and manager terminal interfaces. Providers and managers will use their corresponding terminals in order to access the ChocAn data processing software. Providers will use their interface to login to the terminal, access ChocAn member authentication services, and record and validate services they've rendered to members. Managers will use their interface to login to their terminal, access weekly service summary and financial reports, and access provider and member management services. Depending on the type of account the user logs in with, the user will be given appropriate access and permissions to the different functionalities of the software.

3.2 Application Validation

The Application Validation layer is where various data authentication and validations take place. Managers will login to the software via the manager terminal by entering their username and password. Using the provider's terminal, a provider will log in with their assigned provider ID. Once a provider is logged in, they may validate their patient's ChocAn membership status before providing services, as well as record and validate the service provided by entering the appropriate procedure code and other service details.

3.3 Application Services

The Application Services layer handles report generation, data import and export, and member and provider management. There are two types of accounts in this software: provider accounts and manager accounts. Managers with the appropriate permissions are able to access the functionalities of this layer, such as adding, removing, and updating member and provider demographics, as well as viewing weekly service summary and financial reports. This layer is also responsible for data imports with collaboration with Acme to update membership status records daily at 9:00PM EST, as well as exports to notify them of new memberships and any updates to member's demographics.

3.4 Database Services

The Database Services layer is responsible for the secure storage of member and provider information. It also stores the record of services provided to members, the providers that have rendered services, transaction information such as fees due to providers by ChocAn, and the Provider Directory information including procedure codes and their corresponding service fees. This data is stored and maintained in an ordered list.

4 System Architecture

This section of the document outlines the software and hardware architecture of the system. ChocAn is a software intensive system, consisting of custom-made code to meet the necessary requirements. This System Architecture section describes the interfaces which components communicate with each other and describes what the class is responsible for.

4.1 User Interface

There will be two different interfaces for the ChocAn data processing software depending on which terminal is being used, either the provider's terminal or the manager's terminal. The provider gains access to service recording and billing features of the software from their terminal by entering in their provider ID. Similarly, a manager accesses their terminal by entering their login ID and their password in order to access a higher level of functionality.

4.1.1 Manual & Electronic Input

There are several data elements that are stored in the system, most of which is classified as personally identifiable information, and a few being health related records. All information being inserted into the database will be managed by standard data validation tools. Data validation will be defined to ensure that the information is fit for field purpose, accurate, and consistent with the data type. The information validation includes checking the data type, range and character constraints, data structure validation. Data-type validation will verify that the characters input are consistent with the correct primitive data type.
Validating the range and character constraints processes the user input and input length, and compares it to the minimum or maximum string length allowed. In addition, the validation will evaluate the string length and determine if any restricted characters were used. The validation will also give user feedback to understand if rules and constraints have been broken.

Data structure validation takes processes that are invoked in data-type validation and adds to the process which verifies the data structure is built correctly. The data structure integrity is checked before any management has been done to it as well as validate when an insertion or removal of a node or data is done.

4.1.2 Provider User Interface

The provider's terminal is a tool that helps a provider record and bill for ChocAn services. Once the system is booted, it will prompt the provider to enter their provider ID using the keys on their terminal. Logging in as a provider will allow the provider to have limited access to the ChocAn system for the purpose of recording and billing for services, as well as validate membership status before beginning their service. This limited access will be determined in one of two application classes that will only include the functionality necessary for a provider to perform these duties. The provider's corresponding application class displays all the necessary functionality for the provider to choose from via their terminal.

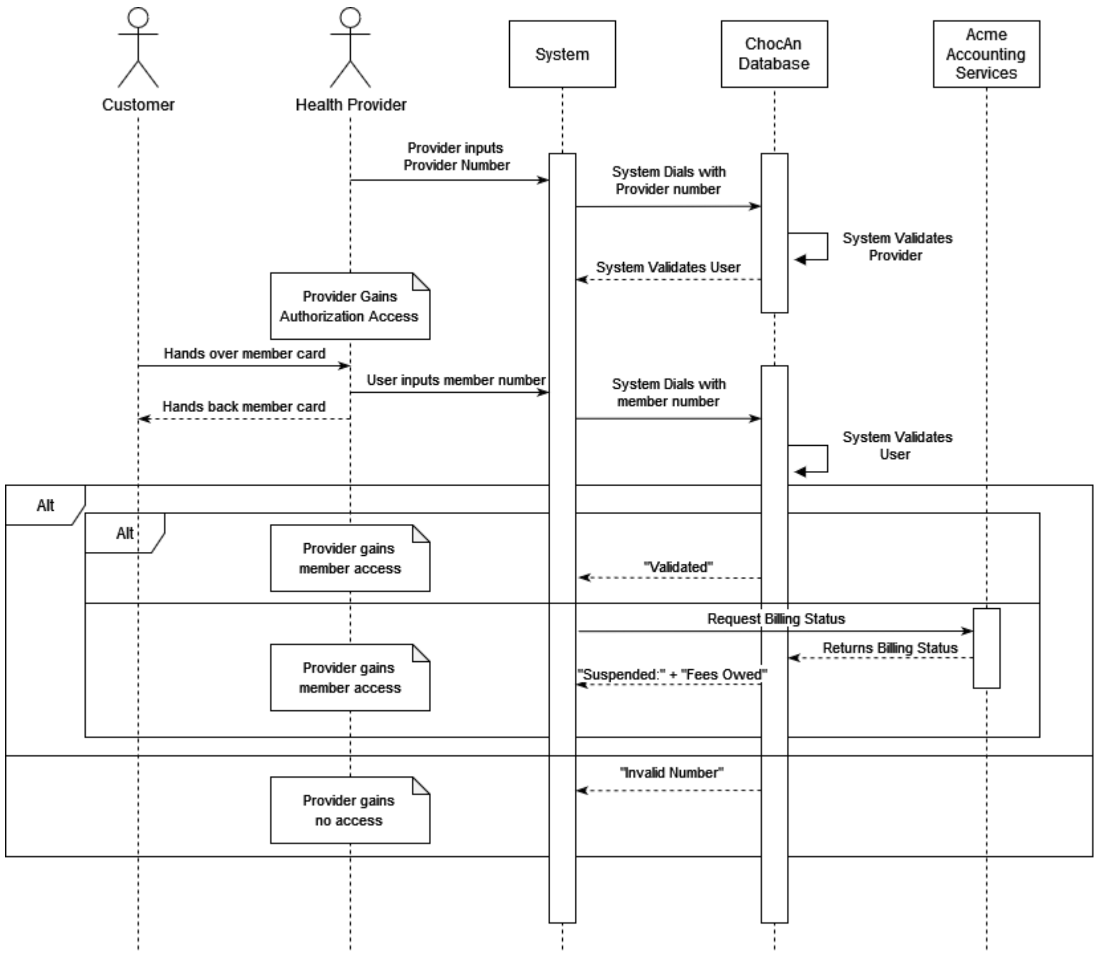
4.1.3 Manager User Interface

The manager's terminal is to be used in order to access weekly reports, such as member summaries, provider summaries, EFT data records, and a summary report for ChocAn accounts payable. It may also be used to add, delete, and update member and provider information. Once the manager's terminal is booted, it will prompt the manager to enter their login ID and password using the keys on their terminal. Logging in as a manager will allow the manager to have the high-level of access to the ChocAn system for the purpose of provider and membership management, as well as taking care of weekly financial procedures. This higher-level of access will be determined in one of two application classes that will have all the functionality necessary for a manager to complete their job duties. The manager's corresponding application class displays all the functionality for the manager to choose from via their terminal.

4.2 Application Validation

4.2.1 Member Authentication

Members who have joined the ChocAn subscription will have a member ID number associated with their account in addition to their personal information. When a member has provided their membership ID number to the provider for the system, the system will request to search for the membership ID number in the database. This is accomplished by searching for the member's ID number iteratively through the database, and once confirmed will have access to the member's ChocAn account. The system will then check for the member's membership status, and if the user has an overdue balance the system will alert the provider. The system will prevent the users (provider/Manager) from making new appointments.


Customer
Health Provider
System
ChocAn Database
Acme Accounting Services
Provider inputs Provider Number
System Dials with Provider number
System Validates Provider
System Validates User
Provider Gains Authorization Access
Hands over member card
User inputs member number
Hands back member card
System Dials with member number
System Validates User
Alt
Alt
Provider gains member access
"Validated"
Request Billing Status
Returns Billing Status
Provider gains member access
"Suspended:" + "Fees Owed"
"Invalid Number"
Provider gains no access

4.2.2 Login

4.2.2.1 User Login

The system is limited to only allow two types of ChocAn employees, ChocAn providers, and ChocAn managers. When a manager logs in they are prompted to enter their login ID and their password. The input field for the password will not be displayed. The system will display a special character such as an asterisk instead of actual content. The password will be encrypted in such a way that the passwords are put through a hashing algorithm, and salting, both of which are stored in the ChocAn database.

4.2.2.2 Encryption, Hashing & Salting

The system will encrypt the password by hashing and salting to make brute-force attacks, dictionary attacks less effective. The system will salt the password with a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) like CryptGenRandom, then the password will be hashed by scrypt, a cryptographic hash function. When salting a hash, the system does not reuse the salt, instead, each password will have its unique salt added.

When storing a password the system will generate a long random salt using a CSPRNG.

The system will then prepend the salt to the password and hash it with a standard password hashing function like scrypt. Both the salt and the hash is stored in the ChocAn database.

When validating the password, the system will retrieve the user's hash and salt from the ChocAn database. Similar to storing a password, the system will prepend the salt to the given password and hash it using the same hash function. The system will then compare the hash of the password typed with the hash from the database. If they match, the password is correct, the user will log in, and an incorrect password, the user will be asked to try again with a limited number of attempts.

4.2.3 User Privileges

As described in 4.2.2.1, there are only two types of user that have access to the system, provider and managers. Providers and managers will have different privileges by the fact that managers will have the same privileges as a provider with the addition that managers have access to manage ChocAn providers and members. This includes the ability to add, remove, and update provider and member's personal information. Providers will only have the privilege to add an after summary report for the purpose of recording and billing for services.
User privileges are provided after checking and comparing the object's type. This can be done by dynamic casting with the help of virtual functions. This validation ensures that the two classes are allowed to select menu options that they are allowed to access. Both user types will have access to the same menu page with the system alerting the user that they are unable to access a menu option if their user privileges doesn't allow access.

4.2.4 Service Validation

After a service has been provided to a member, a provider will then create a service summary to record the service. The service summary process will require the user's membership number and the procedure code. The procedure code will be cross referenced in the ChocAn database, which is uploaded from an external file. Similarly to the member authentication process, the string will be searched iteratively through the list and once found it will display the corresponding name of the service number. The provider will have the ability to decline if the procedure name was not the one intended to be billed by the provider. In addition, if no service name is associated with the number then it will alert the user that the procedure code doesn't exist within the Provider Directory.

4.3 Application Services

4.3.1 Data Management

All data will be stored in a linked list data structure as well as an external file. This includes all personal information about members and their membership status, as well as information about the providers and their services. The external file will be accessible anytime the system is down and the list data structure cannot be directly accessed. It contains the member list, provider list, and provider directory, and is updated every time there is a change in information (updating addresses, adding services, adding members, etc.). When the system is down, no data can be edited, removed, or added. It can only be read through the external file.

4.3.2 Import and Export Files

The system will always update and export a new file of given information. The ChocAn database will contain the most updated information about members and providers, with an external file being the most recent snapshot of the system. This snapshot will also be a backup if the system database or data structure are corrupted.

The system will work with three data structures, members, providers, and provider directory. Each of which are handled in similar ways, when data structures are exported to an external file, the system will format the information in such a way that it can be then parsed and subdivided into corresponding fields when imported into the database.

4.3.3 Generating Weekly Reports

In the ChocAn database, each provider object contains a list of services they have provided which includes the member information. The system will write the provider information first, followed by the list of services provided chronologically into an external file in a readable format. This process is continued until the list is completed. The weekly reports cannot be imported into the system.

4.4 Database Services

The member database will be sent to Acme every week for them to send out billing information to members. The database is also used by managers to update information about members and providers, and by providers to verify information about members and their membership status. It will return reports, provider directories, and weekly summaries for managers to review. The database will send out weekly reports to members with information about their services, as well as send a weekly report to providers that have rendered services for that week.

4.5 Programming Language

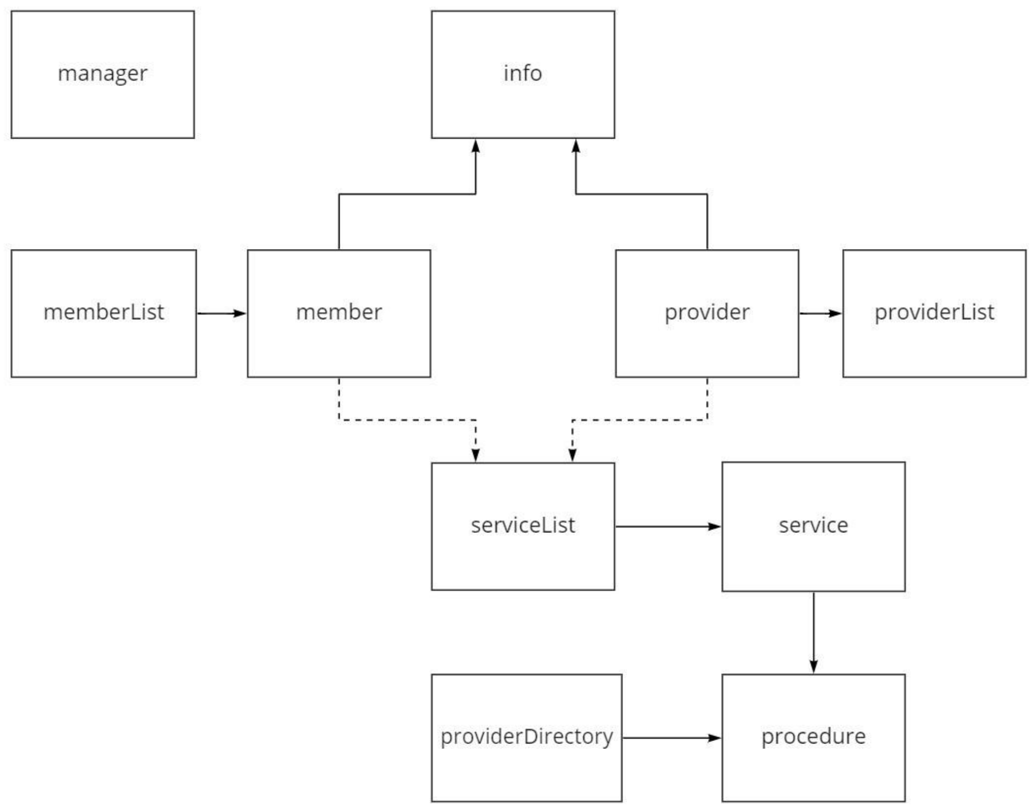
Developers will be using the IDE Visual Studios, creating the software in the programming language C++. This language is well suited for object-oriented programming, which will be at the core of the software development for this system.

4.6 Development and Version Control Environment

Git will be the version control system that is used for software development.

5 Detailed System Design

5.1 UML Overview



The ChocAn data processing software will be structured around the concept of object-oriented programming with a single-inheritance hierarchy. This structure will include the following classes: info, manager, member, memberList, procedure, provider, providerList, providerDirectory, service, and serviceList. In the above UML diagram, the solid arrows indicate an inheriting relationship between classes and the dashed arrows symbolize a containing relationship between classes. Each class and their corresponding variables and methods are explained in further detail throughout this section, as well as their relationships with other classes in the hierarchy.

5.2 Info Class

The info class is responsible for providing the functionality and variables that handle both the member and provider demographic information. This class acts as a base class to its two derived classes, the member class and the provider class.

5.2.1 Info Variables

The info class will include identifying information of both providers and members. This class will include variables such as ID number, first and last name, address information (street address, city, state, and zip code), and email address.

```
class info {
  ...
 protected:
        string firstName;
        string lastName;
        string address;
        string city;
        string state;
        string zipcode;
        string email;
 };
```

5.2.2 Info Functions

This class is responsible for handling any changes that need to be made with regards to provider and member demographic information. Information such as first and last name, address, and preferred email address can be changed if requested and this change request is carried out only by a ChocAn manager. Certain information is unable to be changed, such as the provider or member ID number, as this information is permanent once it is assigned. This class is also responsible for displaying demographic information and inputting in new demographic information if the user logged into the terminal is a manager.

```
 class info {
 public:
      info();
```

```
    ~info();
    info(int ID,
      const string& firstName,
      const string& lastName,
      const string& address,
      const string& state,
      int zipCode,
      const string& email);
    void display() const;
    void read();
    bool change_name (string& newName);
    bool change_address(string& newAddress, string& newCity,
string& newState, int newZip);
    bool changeEmail(string& newEmail);
    bool isUniqueID(int ID);
...
};
```

5.3 ChocAn Member Class

The ChocAn member class is derived from the info class. This class provides the functionalities and variables that are related to ChocAn members, including methods that aid in members receiving their weekly service report.

5.3.1 Member Variables

The members' class inherits information that will identify themselves as individuals from the information class. This includes variables such as the member’s full name, their unique membership ID, address information (street address, city, state, and zip code), and email address. Member status will be determined through a boolean variable which flags when a user is a part of an active ChocAn subscription. The monthly membership fee belonging to the member is also contained in this class.

```
class info {
 ...
protected:
    string firstName;
    string lastName;
    string address;
    string city;
    string state;
    string zipcode;
    string email;
};
```

```
class member: public info {
 ...
protected:
   string memberStatus;
   float memberShipFee;
   serviceList memberSummary;
};
```

5.3.2 Member Functions

The member class handles data that is specific only to members. Once a change of information request has been received, a manager can access the system and update their personal information which include: their name, address, and email. From this class, the system can display the services that members received for that week along with a weekly summary report that is saved in an external file and sent to the member via email. There will be a boolean function that validates whether a membership is valid or not. The member class will hold functions to create and store new member information. A boolean function will ask the user if they want to create a new member, or access an existing member account.

```
    ...
class member: public info {
public:
    member();
```

```
~member();
member(const member & source);
member(int ID,
   const string& firstName,
   const string& lastName,
   const string& address,
   const string& state,
   int zipCode,
   const string& email,
   const string& memberStatus,
   float membershipFee);

void display() const;
void read();
void display_summary();
void save_summary();
void change_status();
void fees_overdue();
...
```

5.4 Provider Class

This class is derived from the info class. The ChocAn provider class will hold functions and variables that are unique to the provider. They will hold information about providers and their services, and have functions that only the provider can access.

5.4.1 Provider Variables

The provider will have a unique provider ID, and a list of services that they offer. A price for each service will accompany the service name. The provider will have their name and area of specialization.

```
class info {
 ...
protected:
    string firstName;
    string lastName;
    string address;
    string city;
    string state;
    string zipcode;
    string email;
};

class provider: public info {
...
protected:
    float weeklyFee;
    int numConsultations;
    serviceList providerSummary;
}
```

5.4.2 Provider Functions

Providers will be able to view the information of the customer they are accompanying with a display function. This would include their personal information as well as the fees owed, and total services provided.

```
    ...
class provider: public info {
public:
    provider();
~provider();
provider(const provider& source);
provider(int ID,
    const string& firstName,
    const string& lastName,
    const string& address,
    const string& state,
    int zipCode,
    const string& email,
    float weeklyFee,
    int numConsultations);

void display() const;
void read();
void displaySummary() const;
void saveSummary();
...
```

5.5 Managers Class

The ChocAn manager class will hold functions and variables that are unique to the managers. It will hold information and functions that only the managers can use.

```
class manager {
public:
    manager();
    ~manager();
    manager(const manager& source);
    manager(const string& managerID, const string& password);
protected:
    string managerID;
    string password;
};
```

5.5.1 Manager Variables

Variables unique to the manager would include the manager's ID number and the password associated with their account. Using these variables, the manager is able to access the ChocAn system and have a higher-level of access than providers.

5.5.2 Manager Functions

Managers will be able to have control over both the ChocAn members and providers for the addition, deletion, and updation of records. Managers will have access to view the summary report given each week that details each active members' expenses and services provided.

5.6 Service Class

The services class will be used to keep track of the information regarding any services recorded and billed in the ChocAn system. This class is derived from the procedure class.

```
class service {
public:
    service();
    ~service();
    service(const member& source);
    service(int procedure_code,
      const string& procedureName,
      float serviceFee,
      const string&
      currentDate,
      const string& DOS,
      int memberID,
      int providerID,
      const string& comments);
    void display() const;
    void read();
protected:
    string mangerID;
    string password;
};
```

5.6.1 Service Variables

The variables stored in the services will be used to keep track of the services provided for each member. That being said, the current date and date of service will be registered. The member and provider ID will be used for verification. The provider will have access to including any comments below 100 characters.

5.6.2 Service Functions

Services should allow for the providers to be able to look and see what services are available through the ChocAn membership.

5.7 Procedure Class

The procedure class will act as a base class to the services class. This class provides the functionality necessary to create procedure objects that will make up the providerDirectory class.

5.7.1 Procedure Variables

The procedure class will hold the unique procedure code, the name of the procedure, and the service fee associated with the procedure. The name of the procedure will be of type string, the service fee will be a float to indicate the dollar amount for the service, and the unique procedure code will be an integer.

5.7.2 Procedure Functions

The procedure class will be able to display its data members and read in a new procedure, if a manager would like to add a new procedure to the Provider Directory. This class can also check that the newly created procedure is being created with a unique procedure code. If the procedure code entered by the manager is already in the system, the system will display an error message to the manager and prompt again for a unique procedure code.

5.8 Provider Directory Class

The Provider Directory class will be used to load in the Provider Directory from an external file, storing this directory into a linked list of procedure objects, and save any updates made to the directory into an external file. This external file can be requested by providers at any time to be sent to their email.

```
class providerDirectory {
public:
    void addProcedure(const procedure& newProcedure);
    void removeProcedure(const procedure& procedure);
    void removeAll();
    bool search(const procedure& findProcedure);
    int getSize() const;
    void displayAll() const;
    void loadFromFile(const string& fileName);
    void saveToFile(const string& fileName) const;

protected:
    int numProcedures;
    list <procedure> procedureList;
};
```

5.8.1 Provider Directory Variables

The provider directory is made of a linked list of procedure objects. This class will also contain the number of procedures located in the Provider Directory to keep track of the size of the linked list. The linked list of procedures will be kept for the external data file to populate the datum.

5.8.2 Provider Directory Functions

The load function will load in any procedures that are already a part of the service from an external data file specific to the procedures into a linked list to where it will get accessed through unique procedure code. This class will also provide the functionality to add, remove, and update any procedures that have been processed through the week.

5.9 Data Structures

The data structures will be created within the system, and managed by the corresponding list manager. Each data structure will contain the following data:

5.9.1 Member List

The member list will be stored in a linked list of member objects, that can be accessed only by the Manager. This class provides the functionality for a manager to add a new member, delete a member, and update member information–dependent upon the updated membership records that ChocAn managers receive from Acme on a daily basis.

5.9.2 Provider List

The provider list will be stored in a linked list of provider objects. Through this class, a manager will be able to add a new provider, delete a provider, and update provider information.

5.9.3 Provider Service List

The provider service list will be stored in a linked list of service objects. Each provider will have a list of their own personal services that they have provided that week. After a service is entered into the system by a provider, it is added to this list. This list can only be accessed by the provider who provided those specific services for the week, and by the manager.

5.9.4 Member Service List

The member service list will be stored in a linked list of service objects. Each member will have a list of the services that they have received that week. After a service is entered into the system by a provider, it is added to this list. This list can be accessed at any time by a manager.