# The ChocAn Simulator

Requirements Document

Table of Contents

[The ChocAn Simulator 1](#__RefHeading__545_1736941561)

[Requirements Document 1](#__RefHeading__547_1736941561)

[1Introduction 3](#__RefHeading___Toc195405503)

[1Introduction 3](#__RefHeading___Toc195405503)

[1.1Purpose and Scope 3](#__RefHeading___Toc195405504)

[1.1Purpose and Scope 3](#__RefHeading___Toc195405504)

[1.2Target Audience 3](#__RefHeading___Toc195405505)

[1.2Target Audience 3](#__RefHeading___Toc195405505)

[1.3Terms and Definitions 3](#__RefHeading___Toc195405506)

[1.3Terms and Definitions 3](#__RefHeading___Toc195405506)

[2Product Overview 4](#__RefHeading___Toc195405507)

[2Product Overview 4](#__RefHeading___Toc195405507)

[2.1Users and Stakeholders 4](#__RefHeading___Toc195405508)

[2.1Users and Stakeholders 4](#__RefHeading___Toc195405508)

[2.2Use cases 5](#__RefHeading___Toc195405511)

[2.2Use cases 5](#__RefHeading___Toc195405511)

[3Functional Requirements 6](#__RefHeading___Toc195405515)

[3Functional Requirements 6](#__RefHeading___Toc195405515)

[3.1Receive Terminal input 6](#__RefHeading___Toc195405516)

[3.1Receive Terminal input 6](#__RefHeading___Toc195405516)

[3.3Compile list at the end of the week 6](#__RefHeading___Toc195405519)

[3.3Compile list at the end of the week 6](#__RefHeading___Toc195405519)

[3.4Provider compiled list 7](#__RefHeading___Toc195405520)

[3.4Provider compiled list 7](#__RefHeading___Toc195405520)

[4Nonfunctional Requirements 8](#__RefHeading___Toc195405521)

[4Nonfunctional Requirements 8](#__RefHeading___Toc195405521)

[4.1Run on Linux (Debian based) 8](#__RefHeading___Toc195405522)

[4.1Run on Linux (Debian based) 8](#__RefHeading___Toc195405522)

[4.2Written in Java 8](#__RefHeading___Toc195405525)

[4.2Written in Java 8](#__RefHeading___Toc195405525)

[4.3Interactive using a Terminal 8](#__RefHeading___Toc195405526)

[4.3Interactive using a Terminal 8](#__RefHeading___Toc195405526)

[5Milestones and Deliverables 9](#__RefHeading___Toc195405527)

[5Milestones and Deliverables 9](#__RefHeading___Toc195405527)

[5.1Refresh of Java language 9](#__RefHeading__783_1736941561)

[5.1Refresh of Java language 9](#__RefHeading__783_1736941561)

[5.2Member visit file creation 9](#__RefHeading___Toc195405528)

[5.2Member visit file creation 9](#__RefHeading___Toc195405528)

[5.3Member weekly detailed list creation 9](#__RefHeading___Toc195405531)

[5.3Member weekly detailed list creation 9](#__RefHeading___Toc195405531)

[5.4Provider weekly detailed list creation 10](#__RefHeading__785_1736941561)

[5.4Provider weekly detailed list creation 10](#__RefHeading__785_1736941561)

# Introduction

This document will outline the requirements for the ChocAn simulator. It is important that this document represents the program attempting to be built so our understanding of the needs of the end user and the challenges of the program is understood ahead of time.

## Purpose and Scope

The purpose of this document is to prep for the next software engineering step in a CS 300 course at PSU. Our scope is a software product to manage a psudo health sciences department in keeping human records data and processing member fees.

## Target Audience

The target audience is the professor of CS300 Fall at PSU Elements of software engineering, Chris Gilmore, and his TBD teaching assistant. As well this is for anyone interested in reading my first piece of software engineering documentation for educational or laughing purposes.

## Terms and Definitions

### ChocAn

Chocoholics Anonymous is a pretend business with a pretend contract we are pretending to take on in order to produce some code with an overall sense of purpose. But without partaking in a business in real life.

# Product Overview

Give a high level description of the functionality of the project here. Describe the purpose of this section. It may be useful to give your definition of a user, a stake holder and a use case. If there are scope limitations to the project, i.e. things you will not be doing, or are not required to do, this is a good section to put those.

The product will be used by employees to serve customers of ChocAn to keep track of patients of ChocAn, keep track of visits, and bill patients.

## Users and Stakeholders

This section will define the people who are interested or have investment in this project. Those who are going to use the product or make money on it

### ChocAn share holder

In the fake ChocAn business a person holding stock in the company will not be involved with this project at all, but will be interested to learn that it exists and helps the company go about its business.

### Gris Gilmore and TA

They will be reading and grading this document and project for quality and overall completeness. They will not be partaking in creating or maintaining code.

### Brian Breniser

As the primary planner and programmer of this software, he will be responsible for creating, documenting, and maintaining the ChocAn code he is assigned for. He will work on the data processing part of the software, but will not focus on the communication, accounting, EFT, or terminal design of the program.

## Use cases

This program will be used in a number of ways, both for pretend reasons and for real reasons

### Chris Gilmore and TA

Will test this code against a set of standards and grade accordingly

### Brian Breniser

Will use and test this code for effectiveness and efficiency.

### Program testing

A program will be used to run this code against test cases and possibly for use in grading.

### ChocAn user

A user of the fake business will need to interact with this program to effectively work in their job. This program will be used to serve users and clients alike.

# Functional Requirements

A functional requirement is the description of the program that specifies what the program will do. It describes behavior or function of the product. Here we will define the pieces of the ChocAn project Brian Breniser will be responsible for.

## Receive Terminal input

Using a keyboard the application must take in text input once it has been run. Our projects portion do not include the terminal front end, but must simulate information coming in from the terminal front end via keystrokes

### File storage of member visit

A file created for every visit a member has to a provider. This file will be stored in the .JSON file format.

### Member visit detailed list

* + - 1. Current date (MM-DD-YYYY),
      2. Current time (HH:MM:SS)
      3. Date service provided (MM-DD-YYYY)
      4. Provider number (9 digits)
      5. Member number (9 digits)
      6. Service code (6 digits)
      7. Comments (100 char) (optional)

## Compile list at the end of the week

Friday at midnight a list containing each member and a list of services for that member, using the member visit file list. This file will be stored in the .JSON file format.

### Member weekly detailed list

* + - 1. Member name (25 char)
      2. Member number (9 dig)
      3. Member street address (25 char)
      4. Member city (14 char)
      5. Member zip code (5 digits)
      6. For each service provided:
         1. Date of service (MM-DD-YYYY)
         2. Provider name (25 char)
         3. Service name (20 char)

## Provider compiled list

This is a list compiled from the perspective of the provider. This list is all the members and services for each member the provider performed during that week. This file will be stored in the .JSON file format.

### Providers list detailed list

* + - 1. Provider Name (25 char)
      2. Provider number (9 digits)
      3. Provider street address (25 char)
      4. Provider city (14 char)
      5. Provider state (2 char)
      6. Provider zip code (5 digits)
      7. For each service provider provided:
         1. Date of services (MM-DD-YYYY)
         2. Date and time received by computer

Date (MM-DD-YYYY)

Time (HH:MM:SS)

* + - * 1. Member name (25 char)
        2. Member number (9 digits)
        3. Service code (6 digits)
        4. Fee to be paid (up to $999.99)
      1. Total number of consultations with members (3 digits)
      2. Total fee fro the week (99,9999.99)

# Nonfunctional Requirements

A nonfunctional requirement is the details of what the program must be, not necessarily do. This will detail the requirements such as OS, language, etc.

## Run on Linux (Debian based)

The program shall run on Linux, specifically a Debian based product

### Can be less precise

As simple as this application is, it will likely run an any Linux distribution, not just Debian or Debian based products, and may run on Unix as well. But this is not a hard requirement so much as a most likely scenario

## Written in Java

The program will be written in the programming language Java, using Java libraries and Java style of programming.

## Interactive using a Terminal

The program will not be interactive using a GUI, just interactive using a terminal.

# Milestones and Deliverables

A milestone is accomplished when a piece of the project, either logically or technologically, is finished, and a new milestone can begin.

## Refresh of Java language

It will take approximately 1-3 days for Brian Breniser to become familiar enough with Java again to start this project. This will likely involve finding his old code and remembering its structure, and building a small side throwaway project for practice. Next is a list of things Brian must learn in order to begin the project

### Accessing the date/time functions of Java

Using built in and efficient libraries find a way to extract the system date and time to put into files

### Working with JSON data

Learn how to create JSON data, store it as a file, load JSON data up and read/write to new files.

### Working with user input

In Java this is pretty straightforward, but a refresh of how to take user input and display output will be needed.

## Member visit file creation

The code that creates a members visit file will be finished on completing this milestone. The writing of code will take approximately 2 days, testing another 2 days, and creating a terminal wrapper around the code for testing approximately a day to create and test.

## Member weekly detailed list creation

The code that creates the member weekly wrapup file will be finished on completing this milestone. The writing of code will take approximately 2 days, testing another 2 days, and creating a terminal wrapper around the code for testing approximately a day to create and test.

## Provider weekly detailed list creation

The code that creates a provider weekly detailed list of services file will be finished on completing this milestone. The writing of code will take approximately 2 days, testing another 2 days, and creating a terminal wrapper around the code for testing approximately a day to create and test.