

# CS Capstone Design Document

May 28, 2020

## Volume-Based Enteral Feeding Calculator

Prepared for

Dr. Judy Davidson

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

Prepared by

Group 60

Enteral Feeding Calculator Team

Alison Jones

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

Parker Okonek

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

### Abstract

The enteral feeding calculator project is an implementation of the enteral feeding catch-up table designed by Dr. Heyland, managed by Judy Davidson, and developed by Alison Jones and Parker Okonek. The project is to create an application that accurately determines a patient's feeding rate based on the time fed and total daily intake. The application should account for missed feeding times and provide the best possible new feeding rate. The application will be developed on either a desktop, mobile, or both environments; depending on user need. This document details the design components of this project. The purpose of it is to increase efficiency and effectiveness among nurses during the tube feeding process. This application will increase the effectiveness and accuracy of nurses. It requires a balance between accessibility and security that creates benefit while maintaining patient security. Many technical components are important in developing this application while adhering to these standards. This document details these technologies and the context they have with regards to users and stakeholders.

# Contents

1	Overview	3
1.1	Purpose . . . . .	3
1.2	Scope . . . . .	3
1.3	Intended Audience . . . . .	3
2	Definitions	3
3	Project Context	3
3.1	Hardware . . . . .	3
3.2	Software . . . . .	4
3.2.1	Desktop Requirements . . . . .	4
4	Design Description	4
4.1	Design Stakeholders . . . . .	4
4.2	Design View . . . . .	4
4.2.1	Users . . . . .	4
4.2.2	Stakeholders . . . . .	5
4.3	Design Timeline . . . . .	5
4.3.1	Gantt Chart . . . . .	5
4.3.2	Items . . . . .	5
4.4	Design Viewpoints . . . . .	5
4.4.1	Context Viewpoint . . . . .	5
4.4.2	Legal Viewpoint . . . . .	6
4.4.3	Composition Viewpoint . . . . .	6
4.5	Design Rationale . . . . .	6
4.6	Deliverables . . . . .	6
4.6.1	Goal . . . . .	6
4.6.2	Stretch Goal . . . . .	7
4.6.3	Out of Scope . . . . .	7
5	Methods	7
5.1	Application Logic . . . . .	7
5.1.1	Approach . . . . .	7
5.1.2	Concerns . . . . .	7
5.2	Interface Framework Integration . . . . .	7
5.2.1	Approach . . . . .	7
5.2.2	Concerns . . . . .	7
5.3	Time-based Input Calculations . . . . .	8
5.3.1	Approach . . . . .	8
5.3.2	Concerns . . . . .	8

		2
5.4	Data Management . . . . .	8
5.4.1	Approach . . . . .	8
5.4.2	Concerns . . . . .	8
5.5	Interface Design . . . . .	9
5.5.1	Approach . . . . .	9
6	Conclusion	9
7	References	9
8	Change Table	10

## 1 Overview

### 1.1 Purpose

The main goal of the project is replace the manual method of calculating volumetric enteral feeding rates with an automated, digital method that results in less errors and takes less time to complete. This document outlines the development path of the project and details the implementation and design on the calculator.

### 1.2 Scope

The calculator will return adjusted feeding rates to accommodate for missed feeding time for a patient from any variety of factors. The user experience will focus on presenting a minimal and easy to read design where the user, a nurse or dietitian, enters in the minimum needed and easiest to derive data to determine the feeding rate and is returned the adjusted safe rate to catch up on daily volume.

### 1.3 Intended Audience

The intended audience of this design document are the sponsors, Dr. Judy Davidson and her team, and the student development team. The document serves as a reference for the student development team to track the product creation timeline and as a representation of the sponsor's needs and product expectations that is verifiable and referable.

## 2 Definitions

Enteral Feeding: A form of tube feeding used to deliver the appropriate nutrients to a patient. This is done through either the mouth, esophagus, or directly into the digestive system through an artificial opening. (Not to be confused with intravenous administration of medication!)

Volume-based tube feeding: The method of calculating hourly rates of tube feeding in an adjustable manner to assure a specified caloric intake daily.

IRB: An Institutional Review Board is a group that has received authority to review, deny, approve, request modification to, and monitor biomedical research in its institution.

## 3 Project Context

### 3.1 Hardware

The software will support first and foremost a desktop platform for both its development and production. The project is developed for desktop and can be completely developed using normal desktop hardware and be expected to run on common consumer desktops. If the project is developed for mobile, it will be able to run on most off the shelf smart phones or tablets.

- 1) System Architecture: i386 or amd64
- 2) Screen Resolution: 1024 x 768 minimum
- 3) Input: Keyboard and Mouse, or on-screen keyboard and pointer

## 3.2 Software

Software requirements for development will involve github and the chosen user interface framework, Windows Form App, whose libraries are required to build and compile the project. The user interface toolkit will not be required for common use by end users.

### 3.2.1 Desktop Requirements

- 1) Version Control: Github
- 2) Operating System: Windows 7 or greater
- 3) DotNet 4.2 or higher
- 4) UI Toolkit: Windows Form App

## 4 Design Description

### 4.1 Design Stakeholders

**Nurses:** Nurses are our primary target user since they will be the ones directly using the application. Nurses conduct all the direct work including setting up feeding bags, doing the current calculations for catch-up feeding rates, and administering food. Nurses also track how much people eat in the electronic health record.

**Dietitians:** These are the people who calculate caloric needs and recommended diet. When the diet requires tube administration, they determine the caloric need, which formula to use, and the feeding rate. Dietitians also monitor response to feeding in the form of nutritional biomarkers/laboratory tests and alter the recommendations for feeding based on the results. Dietitians are a secondary end-user of our application.

**Physicians:** Physicians order the nutrition, usually based upon a recommendation by the Dietitian. They are the people who check that the prescription and administration are performed accurately. They effectively oversee the process.

Some key stakeholders in our project include, Michael Mercer, a dietitian in charge of dietitian standards, Patricia Graham, a nurse lead providing us lots of insight on how nurses do their jobs and what they might want or expect from our application, Dr. Heyland, the person who invented the original feeding rate catch-up chart; We are effectively automating the usage of this chart and turning it into an application, and Judy Davidson, the nurse and research scientist overseeing our project, providing resources for our success.

### 4.2 Design View

#### 4.2.1 Users

The enteral feeding calculator is a nurse's tool to easily retrieve an adjusted feeding rate. For nurses, the end users, using the calculator will be a simple and time-efficient method when compared to a paper table or manual calculation. The user-facing interface for the calculator will have a minimal number of inputs needed to reliably calculate the adjusted feeding rate to decrease possible input errors and promote common usability principles.

A simple interface lends itself well to both learnability, how easy it is for new users to learn the interface, and efficiency, the speed of task completion when a user has learned the interface. The interface will also feature either a prominent calculation button or automatic result return field to aid in efficiency and readability.

#### 4.2.2 Stakeholders

Our key stakeholders (Judy Davidson, Mike Mercer, Dr. Heyland, and Patricia Graham) are mostly concerned about the technical side and implementation of the project. The main goal of this project is to create an application that will increase the accuracy of the administering nurses; avoiding mistakes made during feeding that can be dangerous for patients. Despite directly influencing the nurses by providing them an application, this also makes the job of dietitians and physicians easier; avoiding additional catch-up/adjustments to the original prescription.

### 4.3 Design Timeline

#### 4.3.1 Gantt Chart

Fig. 1. Gantt Chart

#### 4.3.2 Items

1. Go over the rate table/ process that nurses go through when calculating rate based feeding or catch up rate for feeding.
2. Research which interface to use. Contact nurses and learn whether a phone or computer application would be better. Also ask about inputs/ first design check at this time.
3. Research App development based on chosen interface.
4. Develop Algorithm/ math and logic side to app.
5. Integrate algorithm into app framework.
6. Add time based input for application.
7. Ensure user input is saved locally on device to preserve data.
8. Work on app display and usability (pretty stuff)
9. Check app design again with nurses/ projected users.
10. Update app design based on feedback. (repeat as necessary).
11. Finalize Application and publish? Research what is needed to do with this.
12. Check whether we can start implementation and research.
13. Final Report and Wrap-up

### 4.4 Design Viewpoints

#### 4.4.1 Context Viewpoint

The application should present a more accessible and easier to use option than the current methods of calculating the adjusted volumetric feeding rate. The user experience should be hassle-free, without any unexpected slow-downs, crashes, or graphical artifacts within the program's control.

A select test group of nurses will be surveyed to determine the optimal platform to improve usage rates and satisfaction. A similar group, or the same group, will also be asked to review paper prototypes of the applications basic UI layout. Once the interface for the program is developed, it will be tested by the developers, sponsors, and target users to verify the app functions as expected and returns accurate results.

#### 4.4.2 Legal Viewpoint

In September of 2019, the FDA released a set of changes to medical software policies due to the 21st Century Cures Act. Due to these changes the enteral feeding calculator on a mobile platform would fall under the regulatory category of "mobile medical app" due to its use as an accessory to a regulated medical device [FDA].

The enteral feeding tube calculator may not be considered a non-significant risk device under FDA laws though could be approved under the app's IRB. The app does supplement the functionality of a pump that needs to have a rate of flow entered to properly feed a patient; however, if the calculations done by the app are considered to be a "simple task" as described in the FDA's Policy for Device Software Functions and Mobile Medical Applications then this would not be the case. The application generates a flow rate that is entered into a medical device without outside verification and for patient safety should have its accuracy and error-rate examined by an IRB.

#### 4.4.3 Composition Viewpoint

The project will be composed of three distinct pieces: a user-facing UI to collect and report data, a system interface to manage local storage and compatibility, and the internal logic which calculates results from user input and keeps the program state.

The user-facing interface presents an easy to read form input for the minimum required information to calculate a result and passes those values to the internal logic, when a result is created the interface will display this to the user as well.

The system interface manages the constant values between application runs that is not subject to regular change, for example daily target volume or time of feed reset, and allocates required resources from the operating system. The internal program logic manages the resulting rate and integrates the other two components of the program with itself and each other. This component will also manage the notifications to the user for the UI, such as missing inputs, invalid inputs, and indicating the new rate has been calculated.

This composition ensures that the UI framework can be replaced in the event of the current choice losing active maintenance or not adapting to the changing software landscape. It additionally separates the program logic from the interface display and system interfaces, reducing latency from calculation and allowing the program logic to remain consistent and constant regardless of other changes.

### 4.5 Design Rationale

- Uniform among all hospital rooms.
- Secure/ isolated from outside influence.
- Easier transition to third party.
- Accessible to many users.

### 4.6 Deliverables

#### 4.6.1 Goal

The final product should be an application that automates and optimizes catch-up rate calculations for nurses and dietitians. We have two possible platforms on which to develop our app: mobile and desktop. The main goal is to complete an app for one of these (depending on most desirable platform) and conduct both simulated and field tests with the final product.

#### 4.6.2 Stretch Goal

An additional goal, if we complete the first application ahead of schedule, is to develop, test, and implement another version for the secondary platform (the lesser desirable option for nurses). This means that we would have both phone and computer applications available,

#### 4.6.3 Out of Scope

Integration with user health records or third party health management system. This would take a long time for approval due to sensitive data and is not necessary for the main function of the application.

### 5 Methods

#### 5.1 Application Logic

##### 5.1.1 Approach

The math driving the calculations is only required to be called on user input events and does not change dynamically throughout the course of its use. The language used for the application varies on the platform used for the project. The calculation function will only be called when valid inputs are received, and the form of the functions will take a data-driven approach. Many languages implement features which allow for an optimized and easy-to-read data-driven implementation of functionality.

##### 5.1.2 Concerns

Software developers will focus open source development effort in projects that are in languages they prefer, and both Python and Javascript remain popular languages with many enthusiastic followers. The language chosen will affect the platform versions that are available, while a language such as C++ can run on many platforms without the need to install an additional runtime per computer, a language such as Python is less often used by common users and would require more stringent packaging and installation management to ensure minimal effort between searching for the program to opening it for the first time.

#### 5.2 Interface Framework Integration

##### 5.2.1 Approach

Bound variables from the user interface will be read by the underlying code. These inputs will be checked for validity and converted into their proper data types. Once converted into the types required for the application logic's calculations, the volumetric rate output is calculated and returned to the user. The interface between the framework and the application logic will be performed using the Model-view-viewmodel design pattern.

##### 5.2.2 Concerns

Software developers on open source projects focus their efforts highly on projects where they have the requisite skill. Model-view-viewmodel is a common design pattern for UI integration in desktop and mobile applications. User Interface designers that wish to improve the design will find familiarity with web design frameworks, such as Vue.js, which uses a model-view-viewmodel method to interface between client and server [mvvm]. Users will experience a lower performance impact compared to other options, as the application will remain static when not in use.



### 5.3 Time-based Input Calculations

#### 5.3.1 Approach

This section is important when reducing the math nurses have to do. Instead of asking for total amount administered or number of hours a patient was fed, we could have nurses simply track as they go. Nurses can tell the app at what time they started and stopped and the rate that the feeding was set to. The app should be able to figure out the current time and use past entries to tell nurses what the future rate should be. In this case, nurses wouldn't even need to calculate the number of hours missed, the app should do it. Depending on the platform being used the method can slightly change, however it will always remain fairly standard. We will grab a timestamp of Day:Hour:Minute:Second for calculations.

#### 5.3.2 Concerns

We want to ensure that the application is as accurate as possible. Will the device have accurate time? Will the application know if times don't line up properly? Additionally, if times overlap or aren't in the correct order, it must tell the user an incorrect input was entered and reject the input.

### 5.4 Data Management

#### 5.4.1 Approach

The app should be able to keep track of data being entered and preserve it until the user clears it for another patient. We may even be able to add the ability to manage multiple patients. The app should also preserve time information so that the timezone and start/stop time don't have to be entered more than once. They should remain static unless the user decides to reset them.

#### 5.4.2 Concerns

The most important rules for our data to adhere to include persistence, accuracy, and security. Persistence ensures that the data being defined (such as starting time, total volume, and intervals) continues to exist even when the application is closed and reopened. This can be important when closing the application or even in an emergency such as a power outage. Accuracy and security ensure that the data remains constant unless dictated by the user and that it cannot be changed outside of the application (by user or third party). This ensures the data is not tampered with and does not change between uses. Below are some data storage examples and their respective benefits and concerns.

Shared Preferences: Great for standard settings, this method of information storage saves strings in key-value pairs. This may actually be sufficient for our application unless we need to start saving objects (which may be possible if we allow for multi-patient management).

Internal Storage: This method of data storage allows information to persist without being viewed or changed by other applications or the user. This is kept hidden from the user and due to this, may be ideal for our application. We only want the user to be able to interact with user, time, and rate data from within the app.

External Storage: This method of data storage is used when the user wants to export information to view or send someplace else. This method is not ideal since the application won't be exporting anything. All data being stored is used by the app and shouldn't be viewable or editable by the user unless within the app. (even then some data is still obfuscated)

## 5.5 Interface Design

### 5.5.1 Approach

The chosen framework is one that makes the application easy to interact with and enjoyable while also reducing the amount of developer time to achieve that goal. A desktop platform will be the main focus of development and framework selection.

## 6 Conclusion

The enteral feeding calculator will first and foremost be an application to quicken and reduce the time of administration time for nurses and quicken responses for dieticians, but the calculator solves a variety of other issues as well. The calculator will offer nurses, its users, a simple and reliable interface to produce their needed values. The design and open source nature of the process will provide administrators, providers, and dieticians an easy to justify and available product. The architecture of the project provides future developers and contributors an easy start on upkeeping the project after its initial creation to ensure the application continues to improve. Utilizing a UI framework for the interface of the project makes an app that is visually consistent with many other applications that also use it in addition to reducing development time and improving maintainability. All of these aspects lend themselves to a enteral feeding rate calculator that is simple, fast, accurate, and a proper replacement for paper tables.

## 7 References

“How to Store Data Locally in an Android App.” Android Authority, 20 Nov. 2017, [www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/](http://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/).

“Top 10 Best Mobile App Development Frameworks in 2019–20.” By Alex Hales, [hackernoon.com/top-10-best-mobile-app-development-frameworks-in-2019-612b95cf930f](http://hackernoon.com/top-10-best-mobile-app-development-frameworks-in-2019-612b95cf930f).

Krishna12. “How to GetCurrent Time without Using Picker.” Ionic Forum, 15 Nov. 2017, [forum.ionicframework.com/t/how-to-getcurrent-time-without-using-picker/112208](http://forum.ionicframework.com/t/how-to-getcurrent-time-without-using-picker/112208).

Zak-DevZak-Dev. “Cordova : Android Device Current Date/Time.” Stack Overflow, 1 Jan. 1968, [stackoverflow.com/questions/46781636/cordova-android-device-current-date-time](http://stackoverflow.com/questions/46781636/cordova-android-device-current-date-time).

Martin Fowler. Presentation Model. 2004. url: <https://martinfowler.com/eaDev/PresentationModel.html> (visited on 11/03/2019).

Naveed Khan. Introduction to Data Driven Programming. 2018. url: <https://www.effective-programmer.com/2018/05/27/introduction-to-data-driven-programming/> (visited on 11/08/2019).

Mike Potel. The Taligent Programming Model for C++ and Java. 1996. url: <http://www.wildcrest.com/Potel/Portfolio/mvp.pdf> (visited on 11/03/2019).

The Qt Company. Supported Platforms. 2019. url: <https://doc.qt.io/qt-5/supported-platforms.html> (visited on 11/03/2019).

FDA. U. S. Food & Drug Administration. 2019. url: <https://www.fda.gov/media/80958/download> (visited on 11/19/2019).

## 8 Change Table

Original	Updated
<ol style="list-style-type: none"> <li>1) System Architecture: ARMv8</li> <li>2) Screen Resolution: 720 x 1280 minimum</li> <li>3) Input: Pointer and on-screen keyboard or pointer and physical keyboard</li> </ol>	Removed
Software requirements for development will involve github and the chosen user interface framework, Qt, whose libraries are required to build and compile the project.	Software requirements for development will involve github and the chosen user interface framework, Windows Form App, whose libraries are required to build and compile the project.
<ol style="list-style-type: none"> <li>1) Version Control: Github</li> <li>2) Operating System: Windows 7 or greater</li> <li>3) UI Toolkit: Qt (other options, see below)</li> </ol>	<ol style="list-style-type: none"> <li>1) Version Control: Github</li> <li>2) Operating System: Windows 7 or greater</li> <li>3) DotNet 4.2 or higher</li> <li>4) UI Toolkit: Windows Form App</li> </ol>
Assuming a multi-platform or desktop implementation, Qt, supports a variety of platforms with similar or identical interfaces with the underlying programming language. The layout and bindings to underlying data will be written using Qt's provided API, and the required libraries will be packaged with the software as allowed by the license for distribution.	Removed
<p>Mobile Interface Framework</p> <p>React Native: A very popular cross-platform option for Android and iOS development. This framework is inclusive and uses JavaScript. This makes it very familiar, especially for our team who hasn't had experience with mobile app development. This may be the easiest option to learn of the three listed here.</p> <p>Apache Cordova / PhoneGap: This framework seems to emphasize the ease of use with peripherals such as the phone's camera and other apps. While this is great, we likely won't need to make use of this. The benefit of this framework is that many versions of an application can be created using the same code base which makes developing for multiple platforms easier.</p> <p>An few examples of what this may look like include:</p>	Removed