Diabetes Diary

David J. Halek

Governors State University

CPSC-4900

Prof. Freddie Kato

Current Date

Table of Contents

[Abstract 4](#_Toc37939738)

[System Documentation 5](#_Toc37939739)

[System Overview 5](#_Toc37939740)

[System Diagram 5](#_Toc37939741)

[System Inputs: 5](#_Toc37939742)

[System Outputs: 5](#_Toc37939743)

[System Processing: 5](#_Toc37939744)

[System Requirements: 6](#_Toc37939745)

[System Implementation: 6](#_Toc37939746)

[Program Documentation 7](#_Toc37939747)

[Program Name 7](#_Toc37939748)

[Program Flow Diagram 7](#_Toc37939749)

[Program Input: 7](#_Toc37939750)

[Program Output: 7](#_Toc37939751)

[Program Description: 7](#_Toc37939752)

[Data Structure or Algorithm Analysis 8](#_Toc37939753)

[Data Structure or Algorithm Name: 8](#_Toc37939754)

[Description 8](#_Toc37939755)

[Analysis 8](#_Toc37939756)

[DB Documentation 10](#_Toc37939757)

[DB Description: 10](#_Toc37939758)

[DB Type: 10](#_Toc37939759)

[DB Tables 10](#_Toc37939760)

[DB Relationship Diagrams 10](#_Toc37939761)

[Forms and Screens 11](#_Toc37939762)

[Known Issues 12](#_Toc37939763)

[Future Enhancements 13](#_Toc37939764)

[Other Resources 14](#_Toc37939765)

# Abstract

Type 1 diabetes can be a tricky disease to treat. In order to make sure insulin doses are correct, glucose readings, how much and what kind of food the patient is taking in, what kind of exercise they are getting, and what their dose histories are like must be tracked. If the information is not tracked in a consistent and complete manner the doctor cannot make the most informed decisions. It’s also helpful to the user themselves so they can see trends and act on their own if they notice a problem.

Besides this being a computer science project, it is also related to the health field in that it’s made for a type 1 diabetic to help manage their disease.

This project will be a desktop application that tracks information for a diabetic user and allows them to track their readings and dosages over time. This will allow them and their doctor to adjust insulin doses and diet. Usually when a patient is first diagnosed, they will be put on minimal doses of insulin so the doctor can see how they react first. Having very low blood sugar is a much more immediate threat than having high blood sugar, so common practice is to start small and adjust based on logs. If a patient tends to be higher at a certain meal each day the carb counts, previous glucose reading, time between readings, and dosages can be looked at so adjustments can be made. Carb counts will rely on the patient figuring out what they are eating and approximate amounts. There are many food databases out there that can be used, and of course food labels and proper portion control are a big part of it.

There will be two desktop applications – the first will be turn the data into a spreadsheet for the physician to use. The main app will have a simple interface for the patient to log information, edit previous entries, and see reports showing how the reading compares to their last seven days and month trends. For example, when a breakfast entry is made, the user can see what the carb intake and glucose reading are relative to those trends. A spreadsheet can be generated via the second application, which will handle reporting. This spreadsheet can be taken to the physician. It will allow for showing a straight log of all information as well as averages for each meal for each week. It will show for each day the glucose readings, grams of carbs eaten, and insulin doses per meal.

The result of this should hopefully be excellent record-keeping, a more informed and responsive physician, and a healthier patient.

An example of how comprehensive records can help is this example: the patient noticed glucose readings going up over a period of time. When she reviewed started reviewing her meals and notes she realized that the sugar free gum she was chewing was actually adding a non-trivial amount of carbs. It turns out that sugar alcohols have to be accounted for (six grams of sugar alcohols translates to three grams of sugar), as well as the one gram of carbs.

I would expect this application to allow the patient and doctor to identify trends quickly and react to anything being out of the normal range.

# System Documentation

## System Overview

This system was designed with my mother in mind. She was recently (late last year) diagnosed with Type I Diabetes, and also has some issues with cataracts and poor vision (as a diabetes complication), and sometimes has a hard time with using a computer. I wanted to make this as easy as possible for her to make entries (and for me to enter old ones), while making it as informative as possible.

The system is made of two programs. The first acts as a log of diabetes info taken throughout the day. It both displays the recent information added, and also allows a comparison of the daily values against the previous seven, 14, and 30 days. The second program generates two different reports. The first is the log of all entries and shows each entry on its own line. The second is a spreadsheet in which each row is a day’s worth of numbers. It omits the food descriptions and notes that are in the first report.

The primary consideration was ease of use. I wanted to display a lot of information in an uncluttered way, and to make it as easy as possible to make entries. For the second application I wanted to make it a matter of as few interactions as possible being needed to generate the reports.

## System Diagram



## System Inputs:

* List files, window forms, and DB used as input to the system
* Database is diabetes-diary.db, an SQLite relational database
* Forms:
  + DiaryEntryForm – Accepts the following inputs
    - Entry Type (breakfast/lunch/dinner/bedtime)
    - Date
    - Time
    - Glucose Reading
    - Base dose of insulin for that meal
    - Insulin adjustment based on glucose level
    - Total dose of fast-acting insulin taken
    - Total dose of slow-acting insulin taken
    - Estimated grams of carbs eaten
    - Description of food eaten
    - Notes
  + EntryLogDatagrid – allows user to select an entry to edit it
  + ReportsPanel
    - Report type – can select the following
      * Daily Log
      * Spreadsheet

## System Outputs:

* Entry form will result in database updates
* Database will be used to update the RecentHistory panel
  + Shows current day vs daily, 7-day, 14-day, and 30 day averages
* Db will also be used to update the EntryLogDatagrid, which shows all the log entries made, by default ordered first by date (ascending), and second by entry type (ascending)
* Reports program will output the following
  + Daily log – a log of all entries, ordered by date (descending) and entry type (ascending). It will show all the entry information, and each entry will be a row in the report
  + Spreadsheet – This will show all the information for a day’s entries on one row of the spreadsheet, excepting for the food descriptions and notes.

## System Processing:

The Diabetes Diary program opens up a window with three sections in it. The first is an entry panel on the left, which allows a user to add or edit entries. On the right side, on the top, is a recent history control which shows the current day’s entries compared to the last 7, 14, and 30-day averages for each entry type. Making an entry that falls within any of those entries will result in the averages for that part being recalculated and displayed. The final section is below the recent history, and it displays log entries with one entry per line, and the lines colored based on the glucose value (red for bad, green for good, yellow and orange for intermediate). Making an entry updates the log. Additionally, the user can select an entry in the log, to bring it up in the entry section, where they can make edits.

The Diabetes Diary Reports program simply allows the user to print one of two report types, the daily log, and the daily summary. There are two buttons the user can select, representing the report type.

The environment requirements are simply a computer with Windows 10. Windows 7 and newer should work, but I do not have a machine to test that on. Internally, because I am using SQLite, I need a driver for it, which is part of the program environment. Everything is self-contained.

## System Requirements:

This is Windows software and requires Windows 10. This requires that .NET framework be installed, which should not be an issue on Windows 10 machines but may or may not be come with other Windows versions.

The installation requires a zip file, which supplies the executable for each program, the drivers and framework, as well as the database. The database is SQLite, and the OS is Windows 10. Nothing else should be required outside of the zip file.

## System Implementation:

The zip file must be available, and it should be unzipped to whatever the location the user wants. The database is included, with dummy data. So, to use the software:

* Acquire the diabetes-diary.zip file.
* Unzip to the location you wish it to be ran from.
* Double-click DiabetesDiary.exe to install and run it
* Double-click DiabetesDiaryReports.exe to install and run it

# Program Documentation

## Diabetes Diary



### Program Input:

* Database: SQLite local file diabetes-diary.db
* Forms:
  + DiaryEntryForm
  + EntryLogDatagrid

### Program Output:

* DiaryEntryForm (Diabetes Diary)
* EntryLogDatagrid (Diabetes Diary)
* RecentResultsForm (Diabetes Diary)
* OutputWindow (Diabetes Diary Reports)

### Program Description:

Fully describe how the program works using an input – process – output format.

* *Include the programming language used*.

## Diabetes Diary Reports

### Program Flow Diagram



### Program Input:

* Database: SQLite local file diabetes-diary.db
* Forms:
  + ReportSelectionForm

### Program Output:

* Output Window
* Output file: diabetes-log (csv)
* Output file: diabetes-summary (csv)

### Program Description:

Fully describe how the program works using an input – process – output format.

* *Include the programming language used*.

# Data Structure or Algorithm Analysis

List (data structure)

## Data Structure or Algorithm Name:

### Description

The main data structures used are the Entry class, and the Entries List, which is a dynamic array of all the Entries: List<Entry> entryLog

### Analysis

1. What are the memory requirements, based off of the general number of elements used?

It is a List of entries (dynamic array), so memory is based off how much info is in each entry. This is mostly static, but the notes and food description sizes will vary. The memory requirements for it are essentially linear compared to the number of entries.

1. What are the pros and/or cons associated with the element?

Pros:

Very simple to use. My DataGrid binds to the list, which can automatically update the log view. Using a list of entries allows me to utilize properties in the entry class, making it really simple to format and change things in the DataGrid.

Cons:

The only real con is that it requires keeping the list of all entries in memory at all times.

1. What is an alternative program element that could have been used?

The other way that it could have been done is by populating the DataGrid by querying the database whenever a change is made. So, if an entry is added, the database is added, and then the DataGrid is forced to update by querying the database.

1. Why is the element appropriate for your project’s use?

Even with a lot of log entries the memory requirements are very low, since it is all text data. In the event that the amount of data becomes an issue, it can easily be converted to limit the number of elements being shown.

# DB Documentation

## DB Description:

The database is named diabetes-diary. It holds information about each entry (which is date, time, type of entry, glucose level, doses administered, food eaten, and any notes), a few things about the user (name, email address), a lookup for the entry types, and it has a table (as yet unused) that can hold information about specific prescriptions, as a reference.

## DB Type:

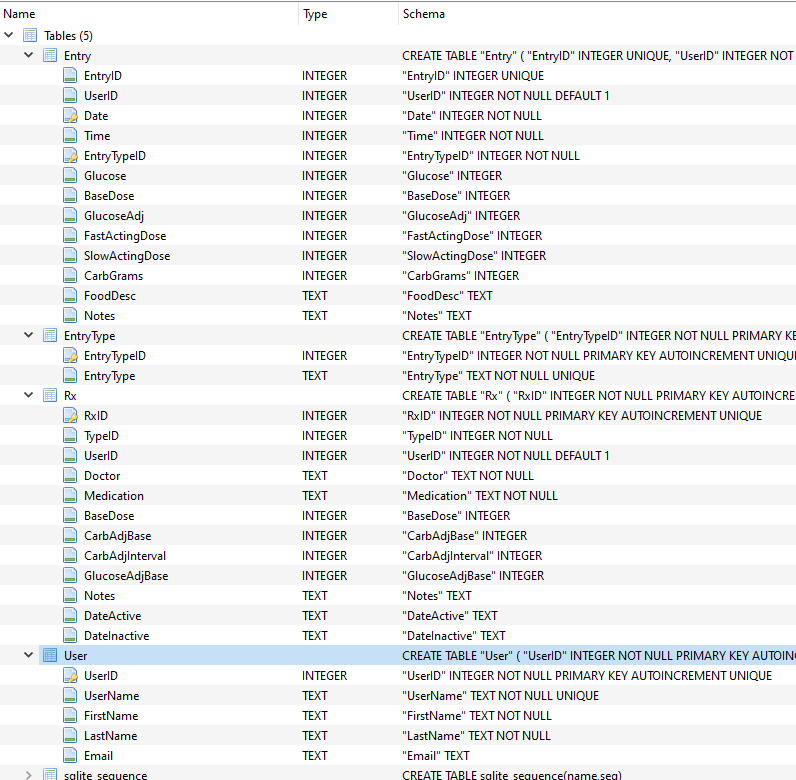
SQLite Relational Database

## DB Tables

* EntryType
  + Lookup table for entry types
  + PK: EntryTypeID
* User
  + Information about the user
  + PK: UserID
* Entry
  + All the information associated with a log entry
  + PK: Date & EntryTypeID
  + FK:
    - UserID -> User.UserID
    - EntryTypeID -> EntryType.EntryTypeID
* Rx
  + Information about prescriptions – Dr, dates active/inactive, specifics on dosing insulin. Not used at this time
  + PK: RxID
  + FK:
    - TypeID->Entry.EntryTypeID
    - UserID->User.UserID

## DB Relationship Diagrams

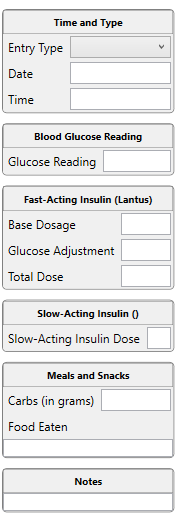




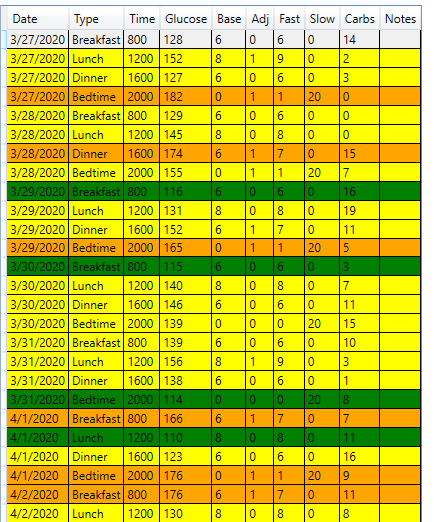
# Forms and Screens

DiaryEntryForm:

This is where the user adds and edits log entries. This can be populated either by selecting an entry from the DataGrid, or by selecting an entry type and entering a date into this form.



EntryLogDatagrid:

This is a list of the log entries made, sortable on all fields, but the default order is by Date (descending) and then by EntryTypeID (ascending). Users can select an entry to have it show up in the entry section. The rows are color coded based on the glucose value given. Green is good, yellow less so. Orange is caution, and red is high/low means immediate and possibly medical attention needed. It also makes it easy to spot trends.

RecentHistory:

This control shows the current day’s entries compared to the previous 7, 14, and 30-day values. This will be extremely useful for identifying trends and for deciding if something needs to be changed, and if so, how to narrow it down. It is updated upon an entry being added or edited.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Today** | **7-Day** |  | **14-Day** |  | **30-Day** |  |
| **Breakfast** | 120.00 | 117.00 | +3 | 119.00 | +1 | 120.00 | +0 |
| **Lunch** | 130.00 | 127.00 | +3 | 128.00 | +2 | 131.00 | +1 |
| **Dinner** |  | 137.00 |  | 136.00 |  | 136.00 |  |
| **Bedtime** |  | 140.00 |  | 141.00 |  | 141.00 |  |
| **Average** | **125.00** | **130.25** |  | **131.00** |  | **132.00** |  |

ReportsMainWindow:

The user can select what type of report to generate – either a log-style report, showing all entries made, with each entry on a line, or a summary report, where each day has its own line. This will result in output to the main window of the chosen report, as well as generating a file.

# Known Issues

1. Date ranges, which will be necessary once there are over 90 days’ worth of entries, and certainly when there is a year or more worth of entries. I need to implement this for both programs.
2. There is an error that does not prevent the program from running, but it shows up on the GUI design screen in Visual Studio. It seems that the latest version of the driver that I am using for SQLite is not always recognized properly by WPF. From researching it seems like it is a known issue, and some people have found solutions, but those did not work for me, and do not seem to work for everyone. It is an obscure problem, but one that I would like to get resolved or at least figured out.

# Future Enhancements

1. It would be nice to implement the Rx entries. This could show how the different doses act over time, plus allow one to see how the prescriptions themselves have changed.
2. A1C tracking. This would mainly allow one to see how A1C has changed over time, and to also compare the average glucose levels over a two to three-month period to the corresponding A1C.
3. Ability to email the forms directly from the application. Since most people do not tend to have a default email program setup, this probably will not be feasible.
4. Allow multiple users, and logins from health care professionals. This is well beyond the scope of what’s practical right now, especially when one considers HIPAA, but it is an eventual goal.
5. Allow user to change their own information. Currently it is just done from the database management software (DB Browser).

# Other Resources

1. SQLite drivers installed via NuGet packages.
2. DB Browser (SQLite)