

# Data Visualization For Patients

Paige M. Applegate

CIS4914, Senior Project  
Department of CISE  
University of Florida

Advisor: Dr. Peggy Borum, *email*: [prb@ufl.edu](mailto:prb@ufl.edu)  
Department of CALS/Food Science and Human Nutrition  
University of Florida, Gainesville, FL 32611

Date of Talk: 7 Dec 2020

## **Abstract**

According to the CDC, approximately 470,000 children nationwide are affected with epilepsy. The Borum Lab aims to help these patients through Precision Ketogenic Therapy (PKT). They help to closely monitor patients' nutritional intake. This includes fats, carbohydrates, and amino acids. Based on how the patient reacts to the prescribed ratio, the team will edit and update the recommended cookbooks. The way that healthcare providers are able to do this is through data analysis. By comparing variables in a graph (whether it be variable vs. variable or variable vs date) and checking the patients z-scores, they are able to see how specific ratios are affecting the patient, and thus making the treatment more effective. The way this used to be done was through microsoft sheets in microsoft teams. While this was a good way to make information accessible to all the providers, it was inconvenient to have to search through and update various folders and files.

To address this, we created web pages for healthcare providers. They would be able to enter data, have that data stored in a database, then be able to view that data through dynamic graphs. The webpage would then be tested by the clinical students to make sure that everything stores/displays correctly and works intuitively. To accomplish this, we use PostgreSQL, Docker, Python, as well as various Python libraries that will be discussed in this paper.

### **1. Introduction.**

In this project we aim to develop a web application that assists healthcare providers in entering data and displaying the information. This particular part of the senior project focuses on visualizing that data for patients. We hope to create graphs that help patients understand exactly what the information is trying to convey. In doing this, the healthcare providers will have a useful tool when showing and explaining data to their patients. This can be done through allowing the healthcare providers to change the graph types, how many variables are on each axis, and be able to zoom into particular sections of the graph.

It has also been noted that patients could have a hard time understanding what the data means past spikes or dips are bad, we also introduced a stoplight feature. This allows

for them to visualize that data is in a range. While variables may go up and down, it does not always mean something is necessarily going wrong.

The schedule for this project is as followed:

08/27/21 - 08/31/21 Identify the Problem  
08/31/21 - 09/10/21 Learn from Databases  
09/11/21 - 09/17/21 Create Wireframes  
09/17/21 - 09/29/21 Create ER Diagrams  
09/22/21 - 10/11/21 Anthropometric Graphs  
10/15/21 - 11/10/21 Clinical Graphs  
10/20/21 - 11/10/21 Other Graphs  
11/16/21 - 11/19/21 Demonstrate Features

Expected challenges are working with the ResVault updating schedule, both when we need to send updated versions of our code in and when ResVault itself is undergoing updates. Another challenge is learning curves. For example, before this summer I had not worked with HTML, Flask, PostgreSQL, PgAdmin, Streamlit, Plotly, or Docker before. I needed to learn them fast and efficiently so that I could use them in my project.

### **1.1. Problem Domain.**

Before working on this project, there was no way for healthcare providers to dynamically update and create graphs. They would have to go into each microsoft sheets file in each of the patients folders to create a graph. They would also have to update each sheet by hand. This process would be time consuming and tedious. Because there is no error checking in sheets either, there would sometimes be errors ranging from not following the data dictionary to simple mistypes. By creating a web page that checks for these errors upon input, this could be solved. The issue with the files could also be solved by creating a dropdown menu that contains the current patients, as well as a drop down menu for the different types of files. With the data being entered this way, it could be stored into a database. The graphs can then pull from this database and therefore be dynamically updated.

### **1.2. Literature Overview**

Through this project, a lot was learned from the Docker documentation [1]. Before running the webpage we had to create, build, and run Docker images. The documentation taught me how to use various Docker commands, be able to troubleshoot Docker related errors, and create Docker images.

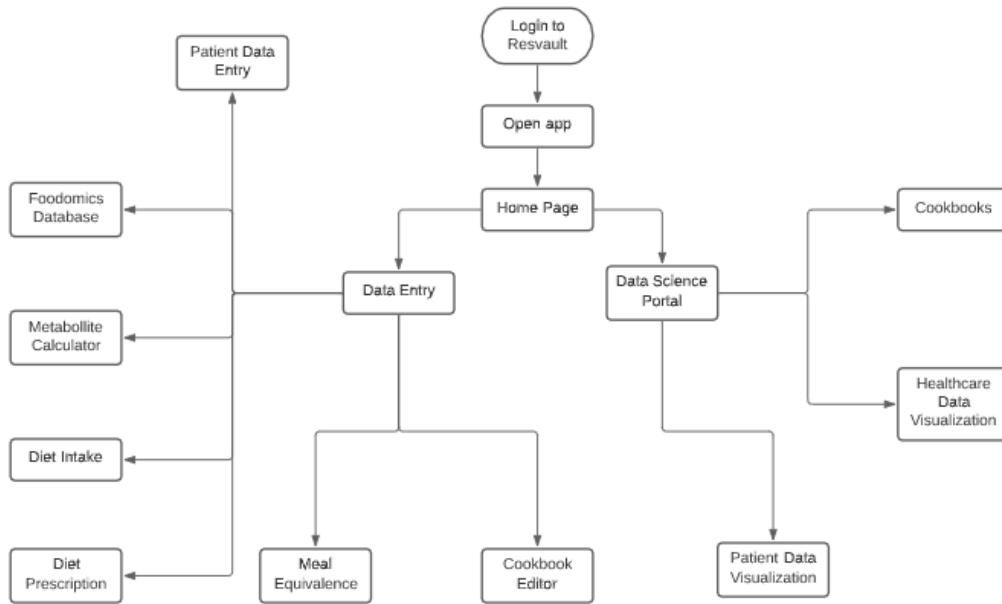
Along with Docker, I had to learn how to use Streamlit [2]. Streamlit is a very useful Python library that was created with data visualization in mind. It is an intuitive platform that allows for the user to create a webpage without having to use HTML or Javascript. After selecting Streamlit as the main platform, I had to learn Plotly from its documentation [3]. It is a very user-friendly Python library that allows for a vast amount of customization of graphs and for the user to download the graphs into a png. The other option we thought about using was dygraphs [4]. However, since dygraphs works in HTML and Streamlit solely uses Python it would be difficult to integrate it into the webpage. The functionality for Plotly also worked better for our end goal.

On the data entry side of the project, I had to learn Flask [5]. Flask allows for the user to transfer from Python to HTML/Javascript and back to Python. It is a good way to create well designed web pages that the creator can customize in whatever way they wish. It is also well known for data entry. Through Flask we could put restrictions on the data being entered, make fields required, and let the user know if there is anything wrong with the way they entered their data.

## **2. Technical Approach (Solution)**

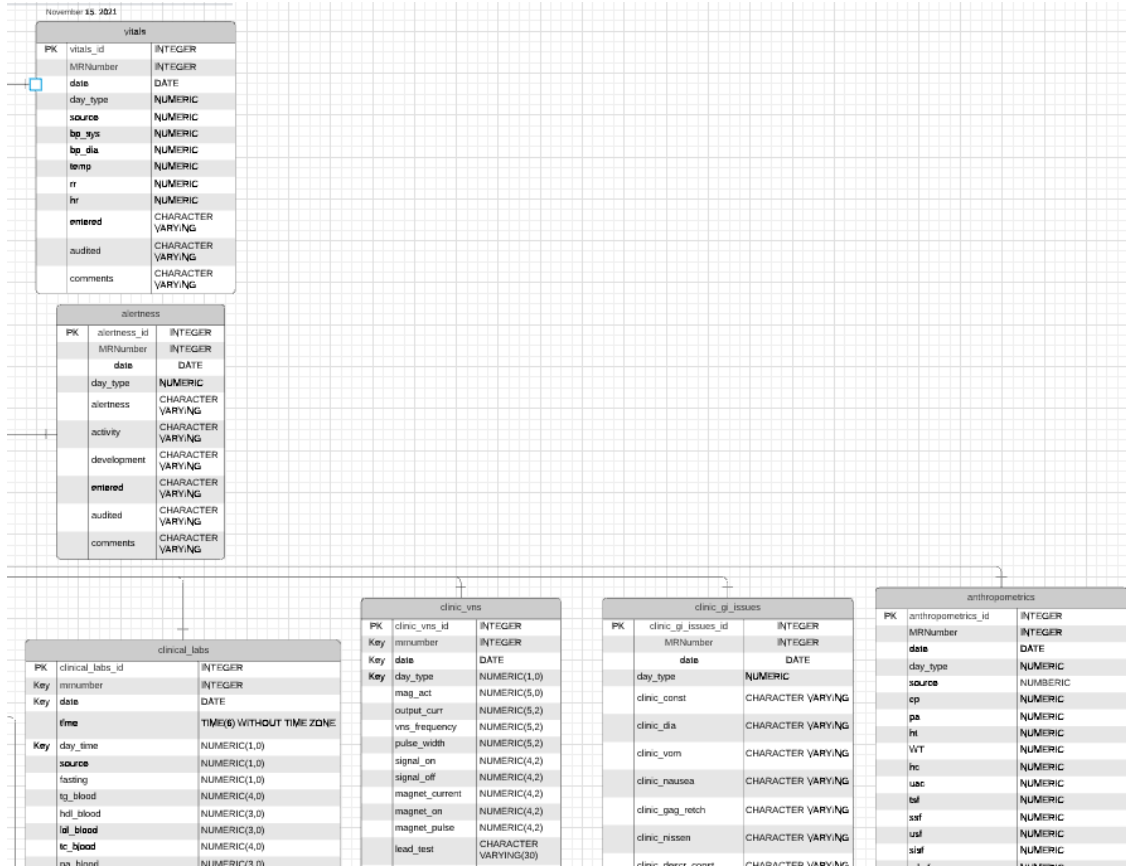
To address the issues faced with only using the excel sheets, we created the patient data entry page and the data visualization pages. To be able to access these pages the user would first need to log into Resvault. Resvault is the application we chose to run our webpage on since it is secure and HIPAA compliant. This is important as we are working with real patient data. After logging in, the user would need to navigate to 'localhost:5000'. This will automatically link to the website's home page. From there they can choose to enter data through the data entry button or look at data output through the data science portal. Figure 1 displays all of the various option available to the user

after selecting one of the two buttons, however the main focus for this paper is the Patient Data Entry and the Patient Data Visualization sections.



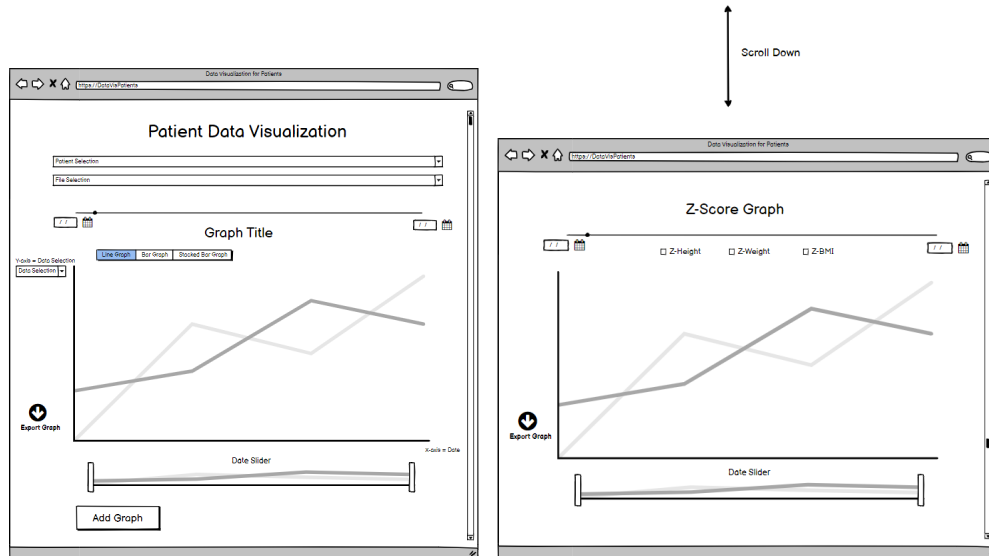
**Figure 1.** Flow chart diagram displaying how the website will work

When selecting the Patient Data Entry, the user is sent to a form. First they must either ‘add patient’ or use the drop down menu to select an existing patient. After that they must select a file they wish to work with. An ER diagram of the available files are shown below in figure 2.



**Figure 2.** ER diagram of the different files. This includes vitals, alertness, clinical\_labs, clinic\_vns, clinic\_gi\_issues, and anthropometrics

After selecting a file, the user can then choose to view existing data or add data. If the user selects add data, they are taken to a form that they can fill out. Each file has their own unique form, each with specific requirements. If they select ‘view’, they are shown a table full of all the current information stored in the PgAdmin database. Next to each row are two buttons: edit and delete. Delete will delete the row selected. Edit will take the user to the file’s form. Any data entered into the form will replace the data in the row selected.



**Figure 3.** Wireframe for Patient Data Visualization page

When selecting Patient Data Visualization, the user is taken to the patient data visualization webpage. In this page the user first selects the patient they want, then selects the file they want to generate the graphs from. The user can then change the graph type using buttons found on top of the graph, export the graph through the export button (which converts the graph into a png they can save on their account in Resvault), and change the date range using the date slider found at the bottom of the graph. If the user wants to change the x-axis values, they have to select 'add graph'. This generates a graph that they can use to plot any value against any other value.

Below these graphs is the z-score graph. The z-score graph allows the user to plot z-weight, z-height, and/or z-bmi against date. The main feature for this graph is the stop light colors. In the z-score graph there are 3 main colors: red, yellow, and green. The green range is  $+1$  to  $-1$ , yellow is  $+2$  to  $-2$ , and red is  $+6$  to  $-6$ . This allows for patients to have a better understanding about what the z-score is trying to convey. Green is the healthy range, yellow is equivalent to so-so, and red is not good. The patient gets closer to the green range, the better the progress they are making.

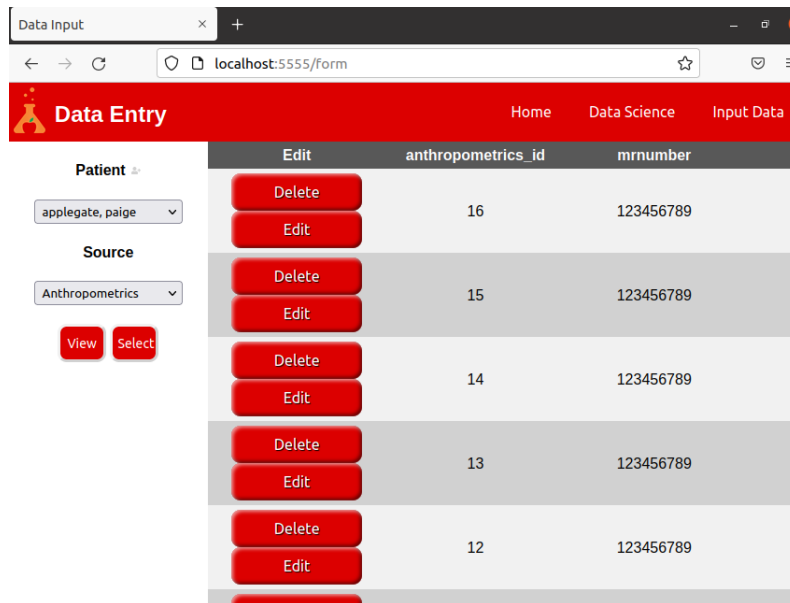
To create the graphs in the patient data visualization section, we utilized PostgreSQL, Streamlit, Pandas, and Plotly. PostgreSQL and Pandas were used to obtain the information

stored in the database (entered by the patient data entry page). From there they were stored into values that Streamlit and Plotly could use. Plotly then graphed the data.

In the patient data entry section, Flask and HTML were used. Flask was used to communicate information taken from the forms created by HTML and stored them into the PgAdmin database. The HTML created the visual part of the webpages, made restrictions on the forms, and obtained information from the user's input.

### 3. Results

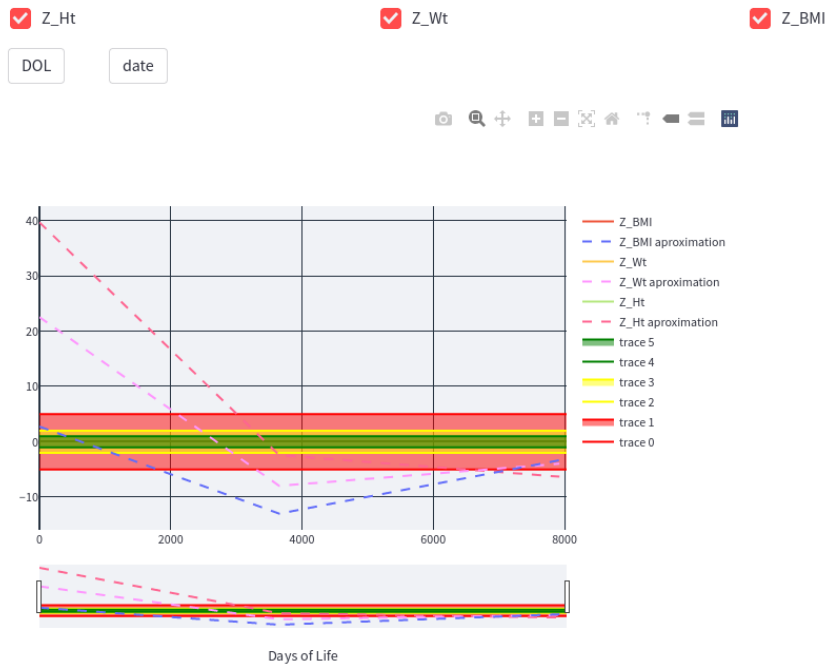
The tests for the webpage ran successfully. Each page loaded up correctly and displayed the appropriate information. Updating the database through the data entry page was successful and all the data stored in the database showed up accurately on the graphs in the patient data visualization page. Results of these pages are shown in figure 4 and 5.



	Edit	anthropometrics_id	mnumber
Delete	Edit	16	123456789
Delete	Edit	15	123456789
Delete	Edit	14	123456789
Delete	Edit	13	123456789
Delete	Edit	12	123456789

**Figure 4.** View tables for the Patient Data Entry pages





**Figure 5.** Z-score graph for the Patient Visualization page

Testing was done through Virtualbox, as well as Resvault. Since I cannot show actual patient data in this paper, I filled the graph with dummy data (an explanation as to why the data looks odd). The stop light function works well in the z-score graphs. A DOL and date button was added to the graph to give the user an option for how they would like to view the data. Normally there would be an option to view the graph by 1month, 6months, 1year, or ytd. Unfortunately, this did not work with the DOL value as it is not in date format. Pressing the date button switches the graph from value vs DOL to value vs date, thus allowing the user to use these buttons.

#### 4. Conclusion

For this project, I created a data visualization page for patients that help healthcare providers explain visually what the data is trying to convey. I also helped in creating the patient data entry page by adding a way to edit already existing data and being able to more easily navigate the web pages. This will hopefully result in the healthcare providers no longer needing to rely on excel spreadsheets to store information.

For future work I would encourage others to add more of the files to the data visualization page as we were only able to get through anthropometrics and clinical. I

would also encourage further testing in the data editing functions. There are simply a lot of values and moving parts in that section, so it is very easy for an error to occur but go unnoticed.

## 5. Acknowledgements

The author would like to thank her advisor, Dr. Peggy Borum, for her guidance, advice, and encouragement towards completion of this project. She would also like to thank Octavio Ochoa for his incredible training as well as Berlin Sankar for her continued assistance with Resvault.

## 6. References

- [1] <https://docs.docker.com/> “Docker (Base Commands)” web:  
<https://docs.docker.com/engine/reference/commandline/docker/> last accessed 15 October 2021.
- [2] streamlit “Streamlit Documentation” web: <https://docs.streamlit.io/> last accessed 29 November 2021
- [3] <https://plotly.com> “Plotly Python Open Source Graphing Library”  
web:<https://plotly.com/python/> last accessed 15 October 2021.
- [4] <https://dygraphs.com> “Tutorial” web: <https://dygraphs.com/tutorial.html> last accessed 29 November 2021
- [5] <https://flask.palletsprojects.com/> “Flask Minimal Application” web:  
<https://flask.palletsprojects.com/en/2.0.x/quickstart/> last accessed 15 October 2021.

## 7. Biography

Paige M. Applegate was born June 6, 1999. She is completing her B.S. degree in Computer Science at the University of Florida (Gainesville, FL). She expects to graduate on December 17, 2021. Through her five years at UF, Ms. Applegate learned plenty about computers and became proficient in C++, Java, Python, and C#. In her free time she enjoyed creating games in Unity, though none of them have been released as of yet, as well as drawing, reading, and playing the flute. She hopes to be employed as a software developer after graduation.