Data Visualizations with Patient Queries for Healthcare Providers

Berlin Somaraine Sankar

CIS4914, Senior Project
Department of CISE
University of Florida

Advisor: Dr. Peggy R. Borum, *email*: prb@ufl.edu
Department of FSHN
University of Florida, Gainesville, FL 32611

Date of Report: 29 November 2021

Abstract

The Borum Lab utilizes various databases, calculators, and other tools to treat patients experiencing epileptic seizures using their Precision Ketogenic Therapy (PKT) program. Of particular importance and relativity to this project is the creation of data visualization outputs for healthcare providers with patient queries. Previously, graphical outputs for the lab had to be manually created in Microsoft Excel; these graphs lacked interactive features and automatic updating systems upon retrieval of new patient data, adding on to the list of tasks that clinical lab members had to perform to procure up-to-date visualizations. Upon reviewing the types of graphs that clinical lab members were providing to external medical staff, the Healthcare Data Visualization project focused on creating a web application in which interactive graphical outputs could be produced for patient progress reports in an efficient, user-friendly manner. Some of these features include date range sliders, graph type selections, dynamic graph creation, and an export feature, among other tools.

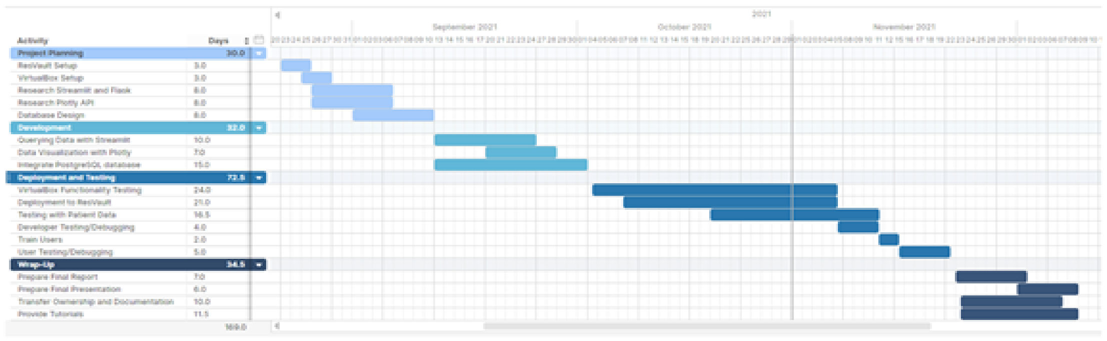
As the Healthcare Data Visualization projects uses real patient data, ensuring all databases and their associated information remained HIPAA compliant was of utmost importance. To develop software for the Borum Lab, the team created our respective projects on VirtualBox with a 64-bit Linux Ubuntu virtual machine. For Healthcare Data Visualizations with patient queries, Streamlit and the Plotly API were used to develop interactive and dynamic graph outputs, database snippets, and a complex query list. The Streamlit application was made accessible from a Flask web page, which was also used for clinical lab members to enter and audit patient data that writes to a relational PostgreSQL database. The application was compressed into Docker images and deployed onto a HIPAA compliant virtual machine hosted by ResVault, a UF Research Computing software used to hold classified information. Upon deployment to ResVault, testing and continued feature improvements were made by developers and with the guidance of the Borum lab's clinical members throughout our development sprints. Upon completion, clinical and computer science lab members were trained to use the developed software to create outputs for healthcare providers and to transfer ownership of the codebase for further development. Software was distributed to Dr. Peggy R Borum and the Borum Lab for continued use.

1. Introduction

The Borum Lab treats patients experiencing epileptic seizures by employing their Precision Ketogenic Therapy program. Previously, the lab relied on Microsoft Excel workbooks to contain their various patient records, calculators, and databases used for PKT treatment. Transferring information between these Excel files had proven to be a difficult process and with increasing necessitation of synchronous, remote work (further bolstered by the Covid-19 pandemic) new software had to be created to procure data visualizations for healthcare providers. Previous patient reports included graphs that were not interactive and had to be individually created by lab members using data that had been manually entered into an Excel database. Over time and continued change of lab members, these databases had become cumbersome to navigate and riddled with formatting errors that made generation of visualizations difficult, resulting in postponement of graph creation for healthcare providers.

However, these visualizations are a vital tool for providing insight into the effects of PKT on a patient. Healthcare providers hold particular interest in them as they allow medical analysis to improve a patient's treatment plan, highlighting data such as how a patient is growing to how they may be experiencing seizures in relation to different medication dosages or diet prescriptions. This is a large motivating factor for creating data visualization software, as it allows for improved treatment of patients and provides the opportunity to learn many data science engineering practices with real-world data. The Healthcare Data Visualization project focuses on using Flask forms to write to a PostgreSQL database that is accessible by Streamlit, where the graphs are created and interacted with. A large portion of the project relies on the use of SQL queries to create complex patient graphs in a way that is insightful to clinical staff in improving patient care. Particularly this is achieved using features such as sliders, dynamic query graph generation, and data entry error handling using Flask forms. My data visualization partner is Paige Applegate who focuses on creating tools for our generated graphs, but with an emphasis on features relevant to patients and their families.

The Gantt chart below (Figure 1) includes the research, planning, development, and testing schedule of the healthcare visualization project. A note of importance is that weekly meetings with Dr. Borum and clinical consultants from the lab were held so that we could perform beta-testing of our developed features throughout the term, allowing us to adjust our software to best fit the activities of the lab.



Activity	Days
Project Planning	30.0
ResVault Setup	3.0
VirtualBox Setup	3.0
Research Streamlit and Flask	8.0
Research Plotly API	8.0
Database Design	8.0
Development	32.0
Querying Data with Streamlit	10.0
Data Visualization with Plotly	7.0
Integrate PostgreSQL database	15.0
Deployment and Testing	72.5
VirtualBox Functionality Testing	24.0
Deployment to ResVault	21.0
Testing with Patient Data	16.5
Developer Testing/Debugging	4.0
Train Users	2.0
User Testing/Debugging	5.0
Wrap-Up	34.5
Prepare Final Report	7.0
Prepare Final Presentation	6.0
Transfer Ownership and Documentation	10.0
Provide Tutorials	11.5
	169.0

Figure 1. *The Gantt chart that was used to schedule research, development, testing, and deployment throughout the senior project course.*

Throughout each development sprint, software was updated to reflect requests from our clinical consultants, allowing for continuous testing and improvement of our software. Prior to deploying software for testing, a major expected challenge was transferring data from Excel to the PostgreSQL database, followed by the ability to integrate the newer Plotly graphs into Streamlit without negatively interfering with the preexisting codebase. Throughout our testing, we were also able to remedy database and visualization errors as they occurred, such as date ranges corrupting graphing tools or the ability to export graphs from the virtual machine to a backed up virtual hard drive.

2. Problem Domain

The Borum Lab intends to improve and tailor PKT treatment of their patients by increasing accessibility and efficiency of their databases, calculators, and other tools. A relational database in which the lab's calculators and patient records could be accessed and used to create visualizations in an interactive manner did not previously exist in the lab. To meet this demand and in the hopes of personalizing treatment plans for PKT patients, the proposed Healthcare Data Visualization software has successfully been able to generate graphs using medical data in a manner that is both HIPAA compliant and intuitive for users. Upon completion and final deployment, this software will enable healthcare professionals to better analyze a patient's performance through a variety of different features that allow treatment adjustments to be made to improve their quality of life. Further features can be added onto this codebase, as well as additional databases, to improve upon this application from both a user and data science perspective, as well as to assist in pursuit of research questions conducted by the lab in the future.

3. Literature Overview

The Healthcare Data Visualization software is intended to work alongside numerous other projects that were previously created for the Borum Lab, as well as to lay a solid foundation for further development to be built upon. Webpages were created using the Flask framework, which supports HTML/CSS code and is widely used throughout the industry to integrate data entry forms into applications [4]. Our team found Flask to be the most intuitive choice as it is easily packageable into a Docker container and can be deployed onto an air-gapped virtual machine [1, 4]. There are many different interactive graph libraries such as Dygraphs [6], but the existing PostgreSQL database was hosted alongside a Streamlit container, so we opted to use frameworks and APIs that could integrate with Streamlit [1, 3]. Streamlit also had an obvious lack of interactive features that were requested for the software, so we turned to other libraries. We also found Plotly to be widely supported by many other software frameworks and libraries, providing more options to future developers [5]. The Plotly API also utilized Python, which meant that the entirety of the data visualization code would continue to be Python.

One of the main concerns when creating the lab's software was transferring everything over to a reliable and robust system that would easily replicate the spreadsheet databases that are currently in use. As the previous databases used by the lab were hosted on Microsoft Excel, it made the most sense to transfer everything over to PostgreSQL database, as this was also the method supported by the ResVault virtual machine. As the ResVault VM is air gapped, PostgreSQL and pgadmin4 were the best fits for the data that the lab would be interacting with for this and future projects. To construct the relational database across multiple Flask applications, we found that PostgreSQL was also easily integrated alongside the existing Flask container. As for Streamlit integration alongside PostgreSQL, it made the analysis of patient data a lot less laborious as the queries easily filtered out patient data via their medical record number instead of the usual process clinical members took of navigating through numerous files on Microsoft Teams and its file system. Another major hurdle was that comparisons of different patient data values was an intensive manual process, especially if values to be compared were on different files when the Excel database was in use. Streamlit and PostgreSQL provided a solution to this issue as we were able to code a method in which only values selected across all databases could be present for its relevant parent category.

4. Solution: Technical Approach

Given that our software would be tested using patient data, our team first had to devise a way in which we could develop our individual projects locally and then transfer it to ResVault in a manner that would not interfere with any of the preexisting software. To achieve this, we utilized VirtualBox to create a local virtual machine in which we could decompress Docker images and run the previous codebase with their respective containers. Using this method, we were able to add our own web pages, database schema, data input forms, and additional containers as necessitated by our project in a way that would be deployable to ResVault. As the Healthcare Data Visualization worked alongside other tables within the PKT relational database, the first technical objective our team sought to achieve was implementing our database in a way that could pull relevant information from previous and soon-to-be-added tables. The tables that are most relevant to the Healthcare Data Visualization portion of the project can be seen in Figure 2.

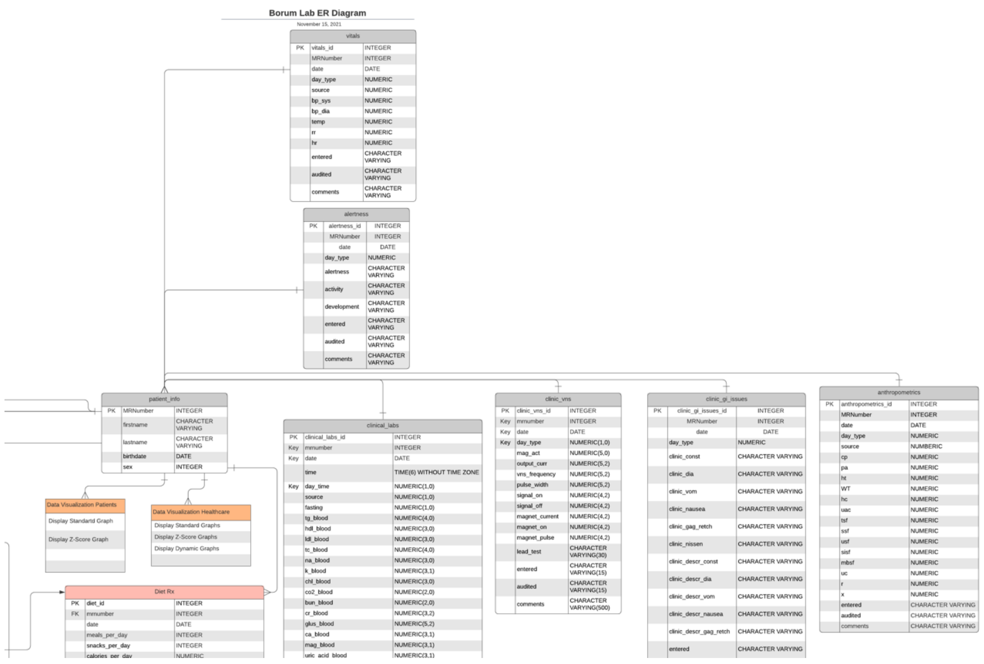


Figure 2. *A snippet of the tables most relevant to the Healthcare Data Visualization portion of the project.*

The software created in this project is intended to serve healthcare providers and other clinical personnel, so creating user-friendly and intuitive webpages were essential to the success of this project. Clinical lab members will be entering in patient data, so it was important to make clear the sequence of actions that a lab member would take to write data to the database through Flask web forms and then navigate to the Healthcare Data Visualization portion of the application. Especially since this project coexists alongside many other calculators and PKT tools, creating individualized portals was key to ensuring successful navigation to and use of the visualization software.

Additionally, as we had routine meetings with our clinical consultants, we were able to get user feedback on our wireframe designs. This allowed our team to develop the Streamlit webpage in a manner that was easy to navigate and provided numerous options for clinical members to generate their graphs. This process assisted in decisions concerning which data visualization tools should be included as baseline features upon first landing on the webpage and which features should be navigated to by other means. The wireframes reflect this feedback, as there is a base graph that contains all interactive analysis, slider, and export features, with additional buttons to navigate towards dynamic graph creation or altering graph type functions. These designs translated over to the actual webpage, as shown in this figure, and our testing has proven this to be useful adjustments for clinical use.

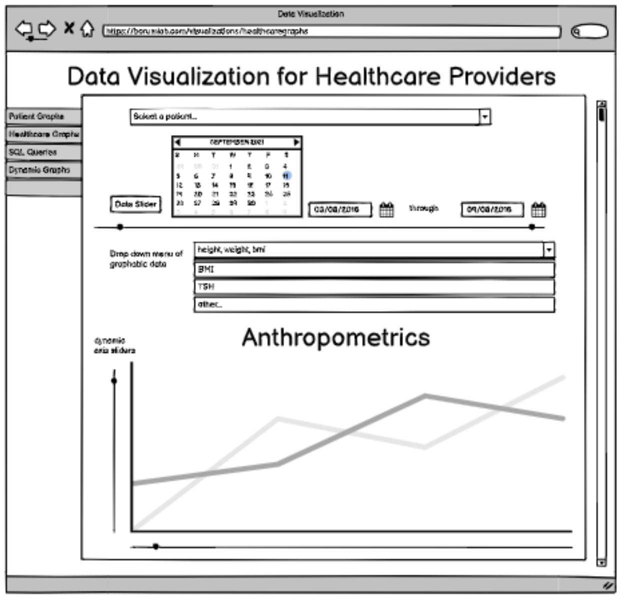


Figure 3. *The proposed wireframe design for Healthcare Data Visualization, including all features requested by the lab.*

Since Streamlit did not have interactive analysis tools that would meet the requirements of the software requested by the lab, our team used the Plotly API to assist in the development of these features. Plotly was largely used to make analysis of graphs easier by creation of interactive tools, and Streamlit was used to create complex patient queries to be displayed by the Plotly graphs. Each graph generated on the Healthcare Data Visualization homepage includes a snippet of the relevant PostgreSQL database for the patient that is selected, as well as the relevant values that have been graphed. Healthcare Data Visualization features of particular interest were sliders to pinpoint certain intervals of time for analysis, the integration of different types of graphs that are generated by complex SQL queries, and the ability to create dynamic axis graphs in which each axis value could be assigned values from in a dropdown. The primary features designed for Healthcare Data Visualizations include:

- *Base Visualizations*: Date vs Y-axis, Bar graphs, z-score queries, export tool
- *Analysis Tools*: Date range sliders/pinch-to-zoom, guardrails, interpolation
- *Advanced Visualizations*: Dynamic axis selection, multiple variable selection

The visualization tools were developed and tested with dummy data, including extreme and erroneous values to test edge cases of the graph generation and additional features. Upon deploying the application to ResVault, patient data was ported into the application, and further analysis tools and advanced visualization features were created to assist in clinical analysis. Due to a recent major update to PostgreSQL, the version used by our application was standardized to maintain functionality of the previous code.

5. Results

The Healthcare Data Visualization software was successful in generating graphs for clinical use and performing analysis on patient performance on PKT. All proposed features, including dynamic graph visualization and requested time analysis tools were successfully implemented for the visualization portion of the project. Accompanying webpages, data entry forms, and database structure was created to improve graphing of queries and ensuring that accurate and correct data was written to the database. Given that patient data is sometimes not accurately recorded or available for lab members to use using the previous Excel database, Plotly’s interpolation tool was used to visualize an estimate of patient values between collected datapoints.

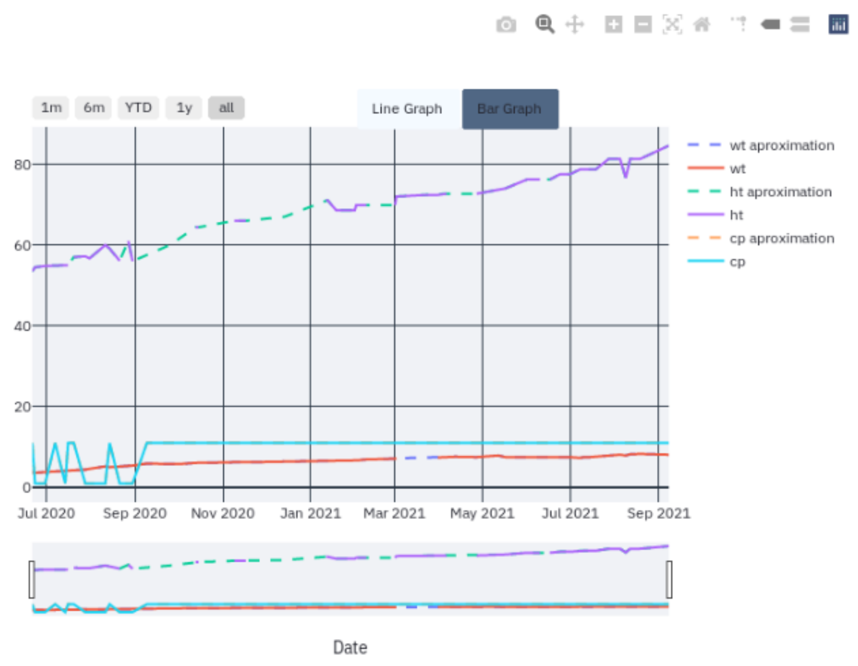


Figure 4. *An example graph generated by Streamlit, showcasing a patient's growth over time.*

All development, testing, and debugging with test data was conducted on a Linux Ubuntu 64-bit operating system via VirtualBox version 6.1.30 with a virtual hard disk of 15.0 GB and 4000 MB system base memory. Flask version 2.0.0, Streamlit version 1.1.0, PostgreSQL version 13.1-3, and Docker version 4.0.1 were used in this project. The application is deployed on ResVault version 1.15.6 by a RedHat Enterprise Linux 7.7 with PostgreSQL (Large) air gapped virtual machine with a resting base network-in speed of 10.27 KB/s, network-out speed of 15.71 KB/s, and 64MB of RAM. Accompanying user drives and data drives at 500 GB each are used to store patient data and host a VM for each user of the virtual machine.

6. Standards and Constraints

Standards: All programming was done in HTML5, CSS 2.1, and Python 3.9.2. *Constraints:* All software was required to run on a RedHat Enterprise Linux 7.7 with PostgreSQL (Large) air gapped virtual machine.

Acknowledgements

I would like to thank my advisor, Dr. Peggy R. Borum, for her continued guidance and advice that extends far beyond the completion of my time in the lab. Additional thanks go to the clinical and computer science lab members I have had the pleasure of working with throughout this past year, their insight and expertise was invaluable throughout this process. I would also like to thank my family for their support and belief in my work, I am continuously inspired by their demonstration of the importance of serving others.

References

[1] "Build your python image," *Docker Documentation*, 26-Nov-2021. [Online]. Available: https://docs.docker.com/language/python/build-images/. [Accessed: 29-Nov-2021].

[2] "Plotly python graphing library," *Plotly*. [Online]. Available: https://plotly.com/python. [Accessed: 30-Nov-2021].

[3] "API reference - streamlit docs," API Reference - Streamlit Docs. [Online]. Available: https://docs.streamlit.io/library/api-reference. [Accessed: 30-Nov-2021].

[4] "Welcome to flask," *Welcome to Flask - Flask Documentation* (1.1.x). [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/. [Accessed: 30-Nov-2021].

[5] "PostgreSQL 13.5 documentation," *PostgreSQL Documentation*, 12-Aug-2021. [Online]. Available: https://www.postgresql.org/docs/13/index.html. [Accessed: 30-Nov-2021].

[6] "Class Dygraph," *Dygraph.* [Online]. Available: https://dygraphs.com/jsdoc/symbols/Dygraph.html. [Accessed: 30-Nov-2021].

Biography

Berlin Somaraine Sankar was born in Pembroke Pines, Florida on March 8, 1999 to immigrant parents from Trinidad and Tobago. She completed her secondary education at Miramar High School and College Academy at BC, where she earned an Associate in Arts degree alongside her high school diploma. Currently, Ms. Sankar is completing her baccalaureate degree in Computer Science with a minor in Business Administration at University of Florida in Gainesville, Florida, where she expects to graduate on December 17, 2021. Ms. Sankar has always had an affinity towards biomedical sciences and technology. Deciding to hone her software engineering skills to improve the healthcare industry, she has become a developer proficient in Python, C/C++, Java, and SQL, with experience in data science and analytics. She has worked with the University of Florida's Borum Lab throughout 2021, creating software to assist with patient treatment, maintaining a HIPAA compliant virtual machine to host the lab's software, and training both computer science and clinical lab members to use these new applications. In her free time, Ms. Sankar enjoys fishing with her family, learning about her Indo-Caribbean roots, and exploring different artistic avenues (her current interests lie in crochet and painting). She hopes to continue to use her skills to improve medical technologies and patient care systems, with the goal of improving accessibility to marginalized and underrepresented communities.