NORTHWEST NAZARENE UNIVERSITY

Creation of the Northwest Nazarene University Finance Reporting Tool

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Ian Wurst

2025

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Ian Wurst
2025

Creation of the Northwest Nazarene University Finance Reporting Tool

Author: Ian Wurst

Ian Wurst

Approved: Barry Myers

Barry Myers, Ph.D., Department of Mathematics and Computer Science, Faculty Adviser

Approved:

Matthew Millsap, Ph.D., Director of Aldersgate Honors College
Second Reader

Approved: Dale A Hamilton

Dale Hamilton, Ph.D., Chair, Department of Mathematics & Computer Science

Abstract

Creation of the Northwest Nazarene University Finance Reporting Tool.

WURST, IAN (Department of Mathematics and Computer Science), MYERS, DR. BARRY (Department of Mathematics and Computer Science).

Gathering and organizing all the financial data produced by the University into an easy-to-read format every month is a very time-consuming process. Not only is it difficult to ensure the data's accuracy, but organizing the information into a format that makes sense to the University's budget managers is also troublesome. The University Finance Reporting Tool allows the University Controller to quickly verify the data's accuracy and generate reports on a much faster timeline. The project started with a set of Excel files that could produce a finished report in a couple of days.

The finished project comprises a backend Python application and a frontend Power BI viewer. The Python application can assemble a full report in less than 30 seconds. It can then be moved into the Power BI viewer in a couple of minutes, leaving the University Controller plenty of time to correct discrepancies. With these results, the project was marked as a success and will be used as the groundwork for future development.

Acknowledgments

I would like to thank my family for supporting me throughout college, particularly my father and mother. I would also like to thank my fellow students, Bryce Miller, Camden Mcgath, Timothy Zink, and Elijah Wurst, for their contributions to the groundwork of this project and their part in forming the person I am today. Last but certainly not least, I would like to thank all of my professors who have spent four years shaping and forming my Christian education here at NNU.

Table of Contents

Table of Figures

Introduction

The student developed a Python application and a Power BI web interface to improve the efficiency of the University Controller at Northwest Nazarene University (NNU). This two-part project and the process of its creation are discussed in the background, project requirements, development, and analysis sections of this paper.

Project Overview

The result of this project is a fully functioning desktop application and web viewer, built to replace and improve some of the work performed by the University Controller in various Excel spreadsheets. The desktop application cuts down processing time by 91% and allows for more time to be spent verifying the accuracy of the information produced by the data provided. The second benefit is the possibility of future improvements being made, allowing for reports to be automatically generated when users need them. The web viewer, Power BI, allows users to easily maneuver the dataset provided by the University Controller and create custom views for specific data points they access often.

Background

The current financial report process is used across campus by budget managers and other administrators at any level of management to identify account balances and transactions when budgeting. John Greentree, Northwest Nazarene University's Controller, generates this report by running a monthly report to retrieve financial data and transactions from the previous month. He generates CSV (Comma Separated Value) files using Evisions Argos reporting tool (Evisions, n.d.) CSV files are generated and pasted into Excel, where Excel formulas enrich the data and prepare the report. Once these formulas are finished,

John manually troubleshoots and reconciles errors throughout the Excel sheets. After most of the errors are resolved, the report is made accessible through the NNU Portal with an interactive, Excel-like interface to view the report and enter data. Budget managers can access the Excel file by accessing the Shared (S:) drive. Some employees prefer only to see specific budgets, sub-funds, or department balances, which can be specified in the portal and generated as a PDF.

There are two main problems with this system. The first is that the University Controller waits a long time for Excel to process the data in the spreadsheets. This is the main reason that John cannot run intermediary reports throughout the month because he does not have the time to wait each time someone wants an up-to-date report. The second problem has been brought to light by the users of the report. Both John and the budget managers want a way to “drill down” and see where a specific number is calculated from. John wants that so he can easily make corrections and the budget managers want a better, more up-to-date way of viewing the budget reports.

Target Audience

The target audience of this project is all budget managers of Northwest Nazarene University. This includes department chairs, sports coaches, program administrators, the University Controller, and the entire finance department. This tool is for everyone who manages, plans, uses, or controls the University's financial resources. Since the current process is slow and tedious for the University Controller to perform and update, this project is designed to be as time efficient as possible for the University Controller.

Project requirements

This project had a few implementation requirements that had to be met throughout the project. First, there had to be some previous data to work with. If no data was provided, then it could not be guaranteed that the system would be able to capture all the kinds of errors present in the data. Second, some form of training had to be provided for the University to learn how to view the reports generated and displayed in Power BI. The plan was to first train the University Controller on how to use the tool, and then after the training, move on to teaching the rest of the budget managers in a series of in-person meetings. The third and final requirement was that further investigation had to be done to determine what Python libraries would be needed for this project. Creating a full and cluttered Python environment is not optimal as it will clutter future work on this project. So, precautions were taken to ensure that no excess bloatware was introduced to the project. One extra note is that during the development process, the necessary documentation steps were taken to allow for ease of access for future work and additions. This sort of documentation includes comments in the code, and various written documents and reading material.

Data entry and software tests were conducted with the University Controller throughout the development and creation of this tool. The University Controller will use this tool, so the University Controller had many opportunities to request edits within the scope of the project. Data entry was limited to Argos reports, while software tests were conducted weekly in a Visual Studio Code Python testing environment.

Development

As a methodology, the student started with several starter analyses of the problem and several ideas on how to approach a solution to the problem. The following sections and subsections explain how this project was developed from start to finish, beginning with an analysis of the project.

Analysis

This project started with prior work done by students for a grade in the Universities Systems Analysis and Design class. The sample starting systems analyses of the project was completed four different ways by teams of three as a semester project. The student combined the projects to come to a final solution. After reviewing the work done by his peers and himself, the student compiled a starting list of three different solutions, narrowing the options down to a combination of Python scripts and Power BI due to constraints placed on the project after its proposal and initial development.

Requirements Gathering

The requirement gathering phase of the project consisted of interviews with the University Controller, three separate budget managers, and a general survey of the University budget managers. Please refer to Appendix A for more details. The interview and survey results of the budget managers were brought to the University Controller who then decided on the final overall goals of the project. Later interviews and meetings were held with members of the IT department to discuss further constraints of the project.

Analysis of the Previous Process

Context

Figure 1, the Context diagram, represents one main process, the budget report system, as a black box in order to focus on its data outflow/inflow relationship to various external entities. The Budget Report System receives financial data from the database, CX/JX. Data errors when running the budget report system are handled by the University's Controller. After corrections are received from the Controller, the budget report system outputs the reports to be consumed by budget managers. Further, the resulting data is stored in the database after the new financial report has been generated and errors have been reconciled.

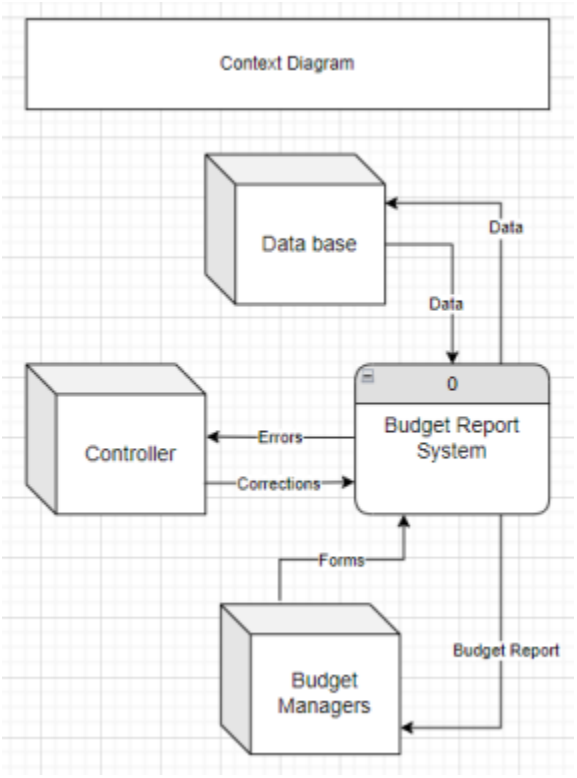


Figure 1: Context diagram displaying the external entities and their relation to the budget report system

Diagram 0

While context diagrams contain one black-box process, diagram 0 looks inside the black box to provide an overview of the processes making up process 0 from the context diagram. As seen in Figure 2, the two main processes in the budget report system are 1) create the budget report, and 2) submit budget report forms. Now there is extra context as to where the inputs and outputs of the budget report system specifically apply. For example, process 1 has the main job of compiling financial data from the CX/JX database into a

budget report for budget managers. It should be noted that, in some sense, not all data flow arrows in these diagrams are equal. Some data flow patterns, like the error/corrections exchange between process 1 and the Controller, are undesirable and essentially side-effects. Identifying these detours and pain points in the data flow helps identify the optimizable components of a system. An interaction with an external entity, the Controller, is identified that is potentially unnecessary as the primary destination of data is the budget managers. In process 2, there are no known undesirable data flow paths or side-effects while the spending request data flows back from the budget managers to the database, and we know that the database and budget managers are the essential input and output entities of the system, respectively.

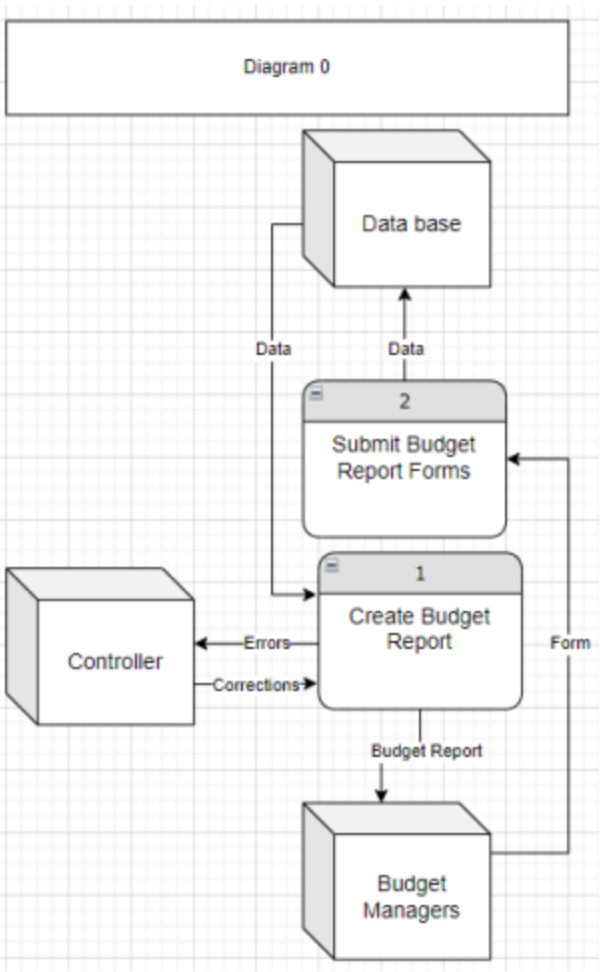


Figure 2: Diagram 0 illustrates the two major process of the budget report system and their interaction with the entities from the context diagram

Lower Levels

The lower-level diagrams are the finest breakdown of processes, breaking down each of process 1 and process 2 from diagram 0 into detailed, disjoint models. Figure 3 outlines in detail the processes involved in creating a budget report from CX financial data. First, Argos pulls financial data from the database, CX, and generates reports that are far from perfect. These reports are typically run monthly on the 15th once the previous budget is finalized. Review of the report brings to light these errors, which are then reconciled by the Controller as modeled in previous diagrams. The final output of process 1.2 is an Excel spreadsheet, which is then added by process 1.3 to the larger Excel data store D1 containing all of the formatted and calculated financial data. The final step is to prepare the report to be viewed through various mediums. Process 1.4 takes Excel data from the data store and generates PDFs, which are uploaded to NNU's interactive portal (process 1.6) and emailed to users who have requested the PDFs from John directly (process 1.5). Process 1.6 also takes Excel data from D1 and directly uploads it to the interactive portal. The final outflow of Excel data from D1 is directly uploaded to a shared drive through

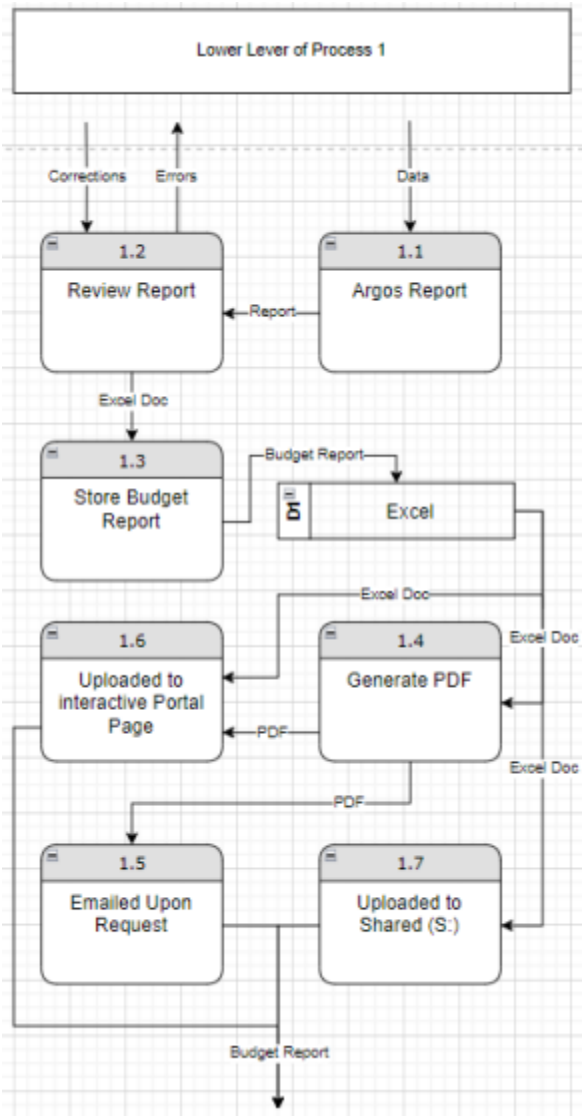


Figure 3: Lower-level diagram of process 1 from diagram 0.

process 1.7 where managers can review the financial report as the Excel file itself. Processes 1.5-7 output budget reports to the budget manager entity as modeled in previous diagrams.

Figure 4 outlines and breaks down passing output data from the budget report forms created by budget managers back to the database from which the Argos report gets financial information. Process 2 is much simpler than process 1 as it simply involves propagating data back to the initial database and input source of the system. After budget managers submit budget requests, those requests are reviewed in process 2.1 and passed along to process 2.2, which in turn enters their spending requests as records in the CX/JX database.

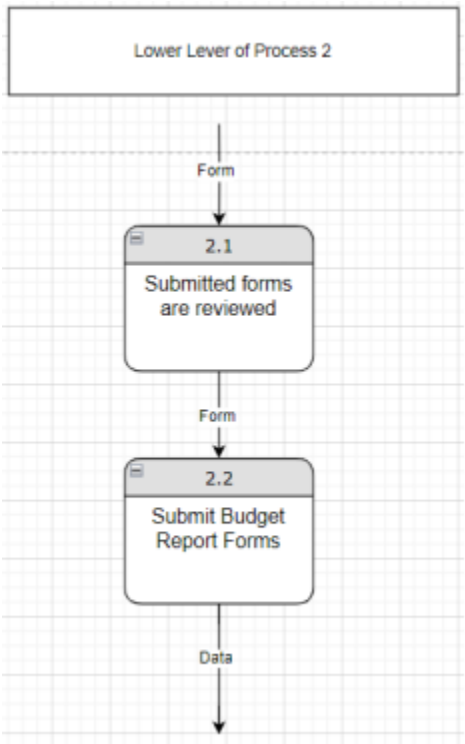


Figure 4: Lower-level diagram of process 2 from diagram 0.

Process Selection

There were three ideas being considered before starting this project to solve the problem at hand. The first was referred to as "Python in Excel." The name is pretty self-explanatory, but essentially its plan was to craft a button using an Excel extension that incorporated Python scripts into Excel cells, allowing for the creation of a button that could process the University's data much faster than Excel could on its own. The second option is being titled "Power BI: One-stop Data Science Shop." Power BI has the processing capability to perform all operations the University Controller needs. The problem with this approach and why it was not chosen was due to the limitations on loading the data into Power BI.

Both options were scrapped in favor of a third option due to new constraints on the project that were discovered after production began. This third option was titled “The University Finance Tool.” This solution was a combination of the previous two ideas. It kept the fast processing of Python and the data visualization of Power BI.

“Python in Excel”

The idea of adding a button inside of the Controller's Excel sheet was the first solution proposed by the Controller. There are two ways to put a button in an Excel sheet that can run a Python script. First is an extension created by Microsoft and is available for the latest versions of Excel (Microsoft Support, n.d.). When this project started in May of 2024 this solution was relatively new and did not have very many features. It works on a cell-by-cell basis and works very similarly to how Excel formulas work. There is no way to make a standard function in this version of Python. However, there are workarounds for that because you can accomplish the same goal of most functions by calculating values in certain cells and then referencing those cells afterward.

The second common way to use Python in Excel works a little differently It is PyXLL. PyXLL is a software solution created by Tony Roberts in 2010 to bridge the gap between Python and Excel (PyXLL, 2024). Instead of working on a cell-by-cell basis and being limited to built-in functions, PyXLL allows people who are more experienced with software development to create their own functions and incorporate them into their Excel documents allowing developers to create visuals and manipulate data. This solution would have worked for what the Controller needed. The downside, however, is that this is not a free, open-source tool. Also, this project had no budget for external software solutions, so it was ultimately scrapped.

In addition to not having a budget for external software, the problem with both solutions is that the University restricts the use of the version of Excel that can run Python scripts. All other external extensions are blocked for security reasons. Other options were considered. The best that still put a button into an Excel file was to use Microsoft Visual Basic for Applications (VBA) to call the necessary Python scripts. While this potential solution was feasible, as it has been done before, it is riddled with complications. Instead of looking for ways to fix those complications, development moved on to the next solution considered, Power BI. Development switched because Power BI was a more feasible solution, and the University already had Power BI.

Power BI: One-stop Data Science Shop

This next solution is unique in its approach compared to the other proposed solutions. Power BI works by creating a database in the cloud and pulling data out by referencing a title given to the data. In this form, mass operations can be performed across the whole dataset. All operations that the University Controller performed on the University's finance data could be completed in the cloud by Power BI. It was estimated at the time that Power BI could have performed all the necessary calculations and manipulations in 3-5 minutes. This is only an estimate, and it should be noted that we were only able to refresh the dataset every half hour, so effectively, it could take as long as 30 minutes. The time to beat for this project was closer to the 9-hour mark, so all things considered, it was a large improvement. All that needed to be done was find a way to get the data from the University's database to the cloud. That was solved by a Microsoft SQL connector on a smaller clone of the database containing only the parts that were necessary for the University Finance Reports (QuantumDot2, 2022).

There were some potential downsides to this solution. The specific license required to do this kind of work was not cheap, and the University was considering dropping the licenses it already had. The developer would have to learn a new scripting language, Dax. This turned out to be a non-issue because it is very similar to SQL. What turned out to be the biggest problem and the ultimate reason why this solution was not used was connecting the University database to Power BI. The request to create the database clone and SQL connection was denied by the University IT team. The request was denied because the University was in the process of transitioning to a new database and did not want to set up something new just to replace it soon after. There was also a data privacy issue that the IT team did not want to risk.

Not all was lost. Though the previous two projects were denied, the lessons learned from being denied for one reason or another came together to form the final solution that ultimately turned into the "University Finance Tool." This solution satisfies all original requirements and fits within the constraints imposed by the University Controller and IT team.

The University Finance Tool

This is the solution that was ultimately selected after the initial project solutions were rejected. The solution has two main parts. A data collection part powered by Python and a data visualization powered by Power BI. The data collector part of this solution utilized Python to read database report files in a pipe-separated format. The various reports were all placed in their own folder for easy access by both the University Controller and the Python scripts.

Now, after the data was collected, there was a decision to be made. The processing of the data into information could be done either by Python or Power BI. The developer decided to do both. Most of the time-intensive number crunching was done in Python, while all processing for visuals was left for Power BI. An example of a step done in Power BI is converting how the negative values are displayed, going from (200.00) to -200.00 so that Power BI's automatic visuals can properly display the numbers.

There are a few major benefits to the University Finance Tool (UFT). First, the UFT can process all the data in 5-11 seconds, depending on how far through the financial year the University is. Second, the UFT gets around limitations regarding reading directly from the database, so it is not heavily restricted. Third, it also does not cost the University anything extra to use this tool because no external software was used outside of what the University was already paying for. However, the downside of not being able to read directly from the database is that the University Controller must manually pull specific reports every month. The University Controller already had to do this step so this did not matter in the end.

Development of Python Script

This section is about the first part of the UFT. This part of the project is focused on creating a table of information that grows as the University moves through the fiscal year. This table is a 51-column report on every account the University has used up to some given month. The resulting report after a full fiscal year of use is a 7000+ row by 51-column table in a format readable by Power BI. The columns are broken up into four sections.

The first section is all the information needed to identify a row and categorize it in the correct ways when viewed by Power BI. This section contains 12 columns. Three of

which are just for error handling and are dropped out of the final “Master Data” file. The other 9 columns contain the Account, Fund, Division, Object, Type, Department, Sub fund, Report Line, and Sub Category. The Account column contains a unique 15 character key for each account, and different sections of the key correspond to different account information. The First two characters are for the Fund code. The next five characters are reserved for the Function/Division code. The next four characters are the Object code. Finally, the last 4 characters are reserved for the Sub fund code. The rest of the columns, with the exception of the Report Line and Sub Category, are mapped descriptions based on the corresponding and/or related section of the account code. The Report Line and Sub Category are mapped differently based on a report the University Controller has to give to the University Board of Trustees. These are not mapped based on the account number but instead on a mapping built by the University Controller separately. This is done to build special reports for certain individuals.

The other three sections of columns are much simpler. These sections contain the fiscal information for the Prior Year, Current Budget Year, and Current Year Spending. Each of these sections has 13 columns containing one column for each month along with a total. These columns are where the bulk of the minor processing occurs, each needing to be rearranged and assigned an account key.

Inputs

This Python application needs a total of seven input files. Six of these files are placed in a month folder. The folder can be named anything, but for consistency and future data tracking the Controller was instructed to name each folder created with this application something that included the month and fiscal year it was run for. So, for

example, “2324 June” would be a good folder name. All these folders are located in a folder on the Controller’s computer backed up to OneDrive. This folder was called “Argos Reports”. Along with the month folders, there is one Excel spreadsheet that controls what accounts all get grouped together for special reports and sorting in the final Power BI app.

The six files that change month to month are grouped into two categories: summaries and tables. The summary reports are database reports that contain the Prior Year, Current Budget Year, and Current Year Spending. The tables are a little different. They contain all the information needed to build the account keys and the descriptions for each section of the keys.

Output

By default, the program will output to the input folder meaning that there is a record held in the folder system of what the University finances looked like at any given point in time. However, there is an option to output to a different folder if the Controller finds a need to do so. For more information on the user interface of the app please refer to Appendix C.

There are two main outputs from the app: an error report and a “masterdata” report. Both reports are in the master data format and contain the same number of columns with the exception of three additional columns in the error report that keep track of different kinds of time-related errors. The error reports are usually very small, less than 20 rows. If the error report contains no errors, the app will notify the user that there are no errors. If the user tries to output an error report when there are no errors, they will just get a blank CSV file. The Master data report file is what can be read by Power BI. It is in CSV format and is usually a larger file of up to two or three megabytes.

Development of Power BI Interface

This section is about how the Power BI interface is designed. This part of the project is focused on delivering the same information that the previous reports delivered in a more user-friendly format. This report was focused on creating two different views of the data set targeting the University Controller and the various budget managers. The Controller needs to sort through the entire University's finances to verify its accuracy and catch any mishandlings so that they may be dealt with quickly. The budget managers needs to know how they are doing in their own department and want quick ways to see what they spent in different categories at various times so that they have an idea of what to request be budgeted for the next fiscal year. Both reports are built off the same data set but are focused on different aspects of the data.

University Controller View

This report is focused on finding inconsistencies in the data set. While a report is undergoing testing and verification, the University Controller can use this report to quickly sort through many different report categories all at once. It is in this report that the data is sorted into different "report lines." These categories are set by the Controller and are grouped in a way that allows him to quickly assess large amounts of the data all at once. There are three ways the controller can view these categories, by Month, Variance, and Total. Month has the spending from each category broken down by month. Totals is more of a summary view and contain all totals needed throughout the entire report. Last is the Variance view, which was constructed to do most of the comparison work, making it easy to sort the report categories by largest or smallest variances. Figure 5 shows an example of

what this report could look like[1].

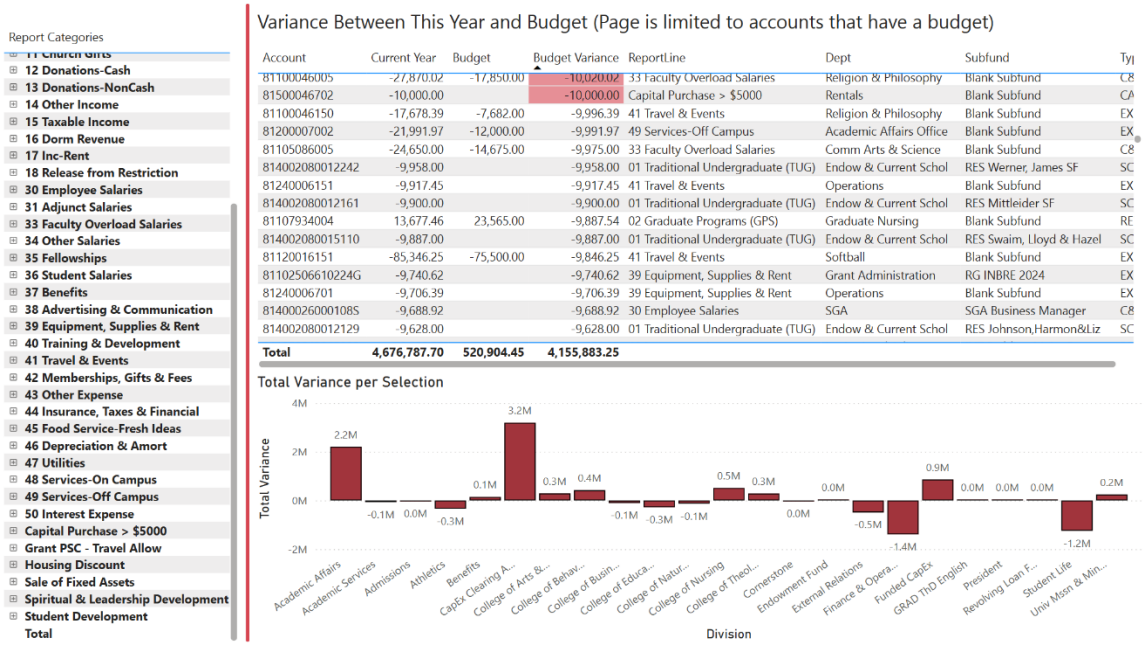


Figure 5: Example of the University Controller View

Budget Owner View

This report was made specifically for the budget managers. It contains all information sorted first by Division and then by Department. On every view of the report, there are pie graphs comparing current, past, and budgeted spending by the selected categories. This report was made so that Budget managers could quickly access information about what they budgeted for a specific item or see what they spent in prior years when they make their budget requests. Figure 6 contains an example of what the report could look like.

[1] Not real data, for educational uses only.

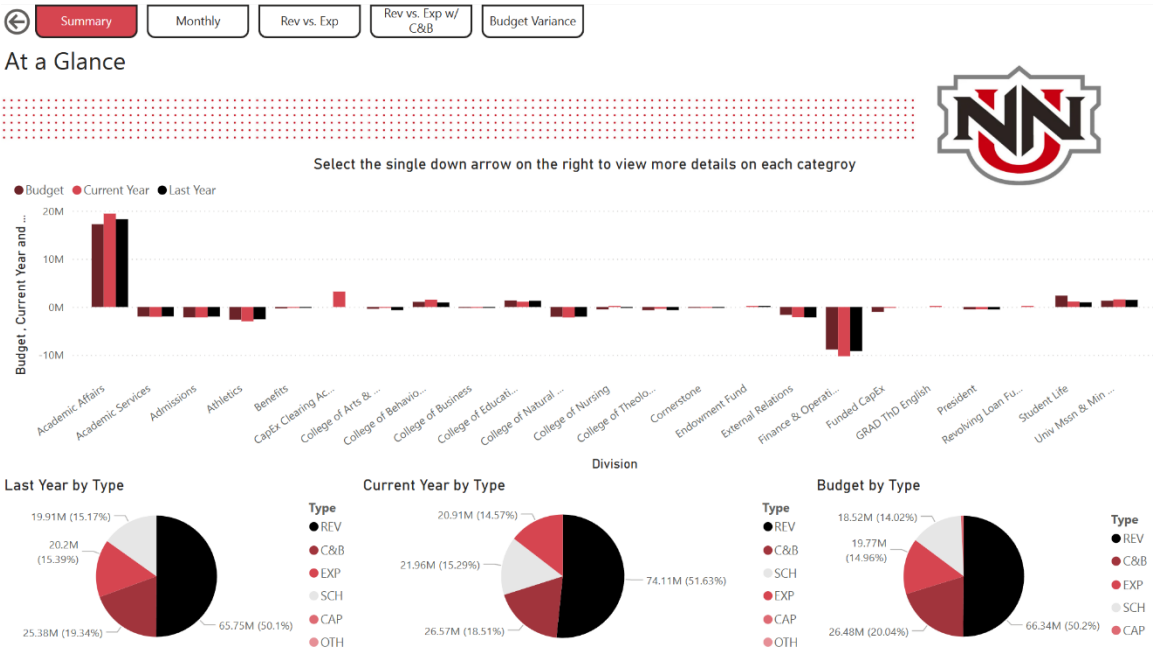


Figure 6: Example of the Budget manager report view[2]

Documentation

The project included an installation guide and basic instructions for first-time use. The installation guide includes information on how to set up the Python environment should any issues arise with the automatic installation. The instructions also contain information about how the folder structure should be set up to get the best results and efficiency out of the project. The same instructions set includes a link to the SharePoint location where the master data file should be uploaded to be viewable by all Power BI reports.

[2] Not real data, for educational uses only.

Future Work

Future work on the project could look very different depending on when the work is conducted. If it is conducted before the switch to Anthology is complete, then direct improvements could be made to the Python application. Those things could be increases in processing speed or UI improvements. Training videos could also be created to allow for easier use and troubleshooting. These training videos would cover how to use the application, how to set up the folder structure for the app, how to use and interact with the Power BI reports, and an introduction to how to make specialized Power BI reports. These videos would be useful both to the University Controller and budget managers.

If the work is done after the switch to Anthology, then the work that has been done here for this project will make a great start, but it will need to be updated to work in the new system. If approval is given by the IT department, then this entire project could be set up to run completely in the cloud using Power BI. This integration could be capable of even more information and data tracking for the University Controller and budget managers.

Conclusion

In the end, the student created a Python application and Power BI web viewer that drastically improved the overall processing time from upwards of 30 minutes to nine seconds. This decrease in processing time allows for more time to verify the accuracy of the data being processed. This results in faster and more accurate reports on the fiscal reports produced by the University. The Power BI viewer allows for an easier step into the role of a budget manager, as you no longer need to know as much about how the University manages and organizes its accounts. Overall, the whole process has been streamlined to make it more accessible and user-friendly for people stepping into new roles at the University.

References

Daniel. (2018, August 2). Python tkinter askopenfilename not responding. Stack Overflow. https://stackoverflow.com/questions/51662441/python-tkinter-askopenfilename-not-responding

Evisions. (n.d.). *Reports*. Reports; Evisions. Retrieved April 12, 2025, from https://webhelp.evisions.com/HelpFiles/Argos/en/Content/Reports.htm

fibonacci_ostrich. (2023, March 9). Terminate tkinter process when "X" button is clicked (python). Stack Overflow. https://stackoverflow.com/questions/75684655/terminate-tkinter-process-when-x-button-is-clicked-python

Gianmarco. (2021, October 25). Filtering by clicking on a visual and maintain across pages. https://community.fabric.microsoft.com/t5/Desktop/Filtering-by-clicking-on-a-visual-and-maintain-across-pages/m-p/2153094#M793034

Gimel. (2009, June 3). How to check for NaN values. Stack Overflow. https://stackoverflow.com/questions/944700/how-to-check-for-nan-values

Gotty. (2019, August 7). tkinter not working when run from command line python script. Stack Overflow. https://stackoverflow.com/questions/57399880/tkinter-not-working-when-run-from-command-line-python-script

Microsoft Support. (n.d.). *Introduction to python in excel—Microsoft support*. Microsoft. Retrieved February 28, 2025, from https://support.microsoft.com/en-us/office/introduction-to-python-in-excel-55643c2e-ff56-4168-b1ce-9428c8308545

Munapo, Joe. (2017, August 26). Tkinter Loading wheel or progress bar. Stack Overflow.https://stackoverflow.com/questions/45896941/tkinter-loading-wheel-or-progress-bar

Nos. (2016, August 16). Python pandas dataframe, is it pass-by-value or pass-by-reference [duplicate]. Stack Overflow. https://stackoverflow.com/questions/38895768/python-pandas-dataframe-is-it-pass-by-value-or-pass-by-reference

Pandas—Dataframe reference. (n.d.). Retrieved February 11, 2025, from https://www.w3schools.com/python/pandas/pandas_ref_dataframe.asp

Python string startswith() method. (n.d.). Retrieved February 11, 2025, from https://www.w3schools.com/python/ref_string_startswith.asp

PyXLL. (2024). *Python in excel with pyxll, the python excel add-in*. Introduction to PyXLL; PyXLL Ltd. https://www.pyxll.com/docs/introduction.html#how-does-it-work

QuantumDot2. (2022, June). *Connect CX Jenzabar JenzaMate CARS system to Power BI*. Reddit.

https://www.reddit.com/r/PowerBI/comments/vnh7oc/connect_cx_jenzabar_jenzamate_cars_system_to/

Riscy. (2011, July 14). How to excute python script when pushing a button on excel. Stack Overflow. https://stackoverflow.com/questions/6688115/how-to-excute-python-script-when-pushing-a-button-on-excel

Shuaki, Esmail Alberto Ghassemi (2022, February 22). Tkinter Scrollbar inactive untill the window is manually resized. Stack Overflow. https://stackoverflow.com/questions/71260406/tkinter-scrollbar-inactive-untill-the-window-is-manually-resized

Tilley, S. R. (2020). *Systems analysis and design* (12th edition). Cengage.

WCTech. (2017, October 30). Tkinter Dynamic scrollbar for a dynamic GUI not updating with GUI. Stack Overflow. https://stackoverflow.com/questions/47008899/tkinter-dynamic-scrollbar-for-a-dynamic-gui-not-updating-with-gui

Wynter. (2019, June 11). Can I save a Pandas DataFrame with a Tkinter File Dialog?. Stack Overflow. https://stackoverflow.com/questions/56549483/can-i-save-a-pandas-dataframe-with-a-tkinter-file-dialog

Appendix:

Appendix A: Interviews and Surveys

Notes from class interview of the University Controller

Q: What is the project's budget?

- John: Depends on the design and what the project is asking for. You have to equate the cost; building a database is different than taking a look at a workflow. Also consider soft cost vs designing a whole database for storage and paying for ongoing cloud and SAS fees. The Project will stay in-house, no outsourcing; Microsoft products can be used if already offered at the University.

Q: What personnel will/could implement the system?

- John: Looking at one paid summer intern to take on this task.

Q: What are the expected and hoped-for ways that the Anthology project will change this system?

- John: The rules that are set up in this current Excel as being able to lay those rules in the Anthology system and you're able to kick off the report using those rules. Hopefully, Anthology will replace all that we're doing and we can build rules and hopefully make it more granular in adding and removing codes. We aren't importing 10 years back into the system, and we need to see trend lines and will need to be able to pull it into Excel until we've been transitioned for almost a decade; trends NOT PART OF THIS SCOPE.

Q: When does the project need to be completed?

- John: Next fiscal year: June

Q: Can we use tools other than Excel?

- John: Open to anything, Excel is the tool I'm able to use now but it is not the final answer and within the soft cost budget, I am open to alternatives to Excel.

Q: What is your primary goal?

- John: Core: Accelerate the timing of this report from close to availability, how can we speed up from when data is ready to when it can be viewed. Stretch: how can we take that data and manipulate and see it in ways that can help enrich how people are getting to answers of why do we have this expense and why is it above budget what is the timing to be able to drill down.

Notes from interview with the Department chair of Mathematics and Computer Science

Notes of interviewer Ian Wurst:

Turns out he has never used the Excel tool and is supposed to be turning in a report for the first time sometime next week. After he got what he needed from John, he was annoyed that he had no way Page 9 to get access to the amount his department had spent already. He asked around and found out that people specifically request this information from John. So, for [him] to be happy with our new system he would want the same information he has now in his budget request document and then the current year to date spending and month spending displayed in an easy-to-read manor. Which it sounds like Power BI does.

Notes from interview with the Department Chair of the Department of Instructional Design and Technology

Notes of interviewer Bryce Miller:

Due to Instructional Design and Technology (IDT) having a very small budget outside of personnel costs (benefits and salaries), [she] almost exclusively uses the pdf of the financial report for IDT that is sent out, and even then, she only reviews it around twice a year.

Notes from interview with the Director of Campus Life

Notes of interviewer Camden McGath:

Q: How do you access the financial report?

- Interviewee: Mostly through the portal and Excel.

Q: What is your favorite thing about the report?

- Interviewee: Being able to sort by object code and sub fund number.

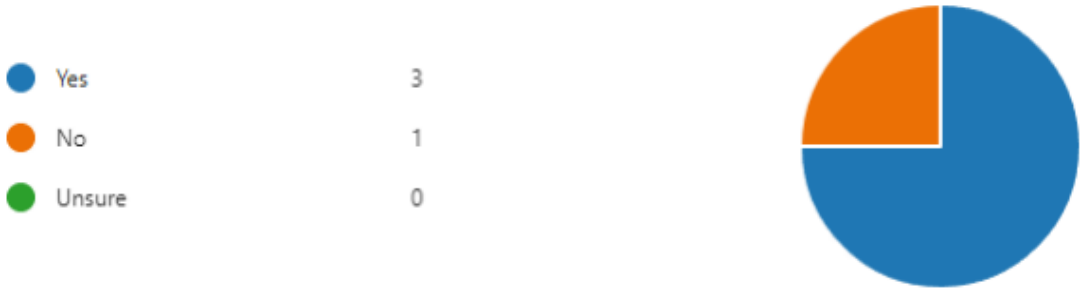
Q: What is your least favorite thing or most wanted feature?

- Interviewee: Want up-to-date financial information, more than last month's numbers.

Survey of Budget Managers Questions and Results

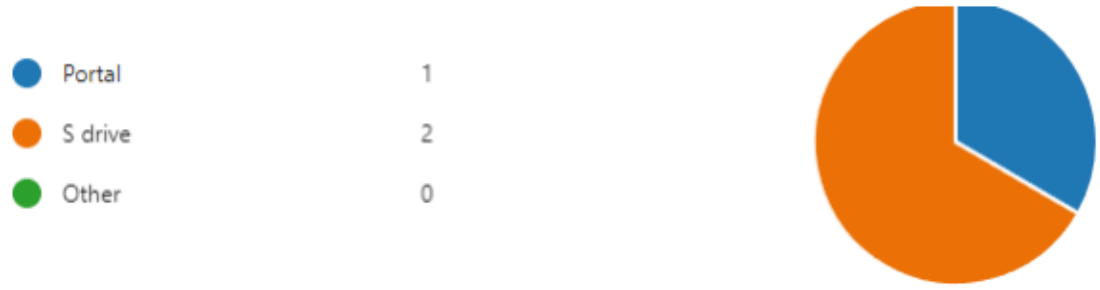
Four survey respondents were from the following departments: Education, HR, College of Business, Senior Leadership Team

Q1: Do you use the monthly financial report?*



*The respondent who answered no was mistaken: later survey answer revealed they do use the report. This respondent was not shown the following questions due to this answer, but did answer yes to using the portal when asked about alternative budgeting means.

Q2: How do you use the report?



Q3: What format do you interact with most?

Q4: What is the most useful or helpful feature of the report?

- Comments and indicator lights
- FY vs Actual dollars spend, YTD spent, remaining dollars or difference of both spent or retained
- The ability to drill down into both the detail behind the summary presentation, and the activity by department/college/office

Q5: What about using the financial report is difficult?

- Remembering if () are good or not
- N/A
- The width of the spreadsheet (necessary to show the detail on how the budget was allocated by month and the prior year's activity by month) is somewhat overwhelming, but can be hidden to make it more manageable.

Q6: What would you change about the financial report?

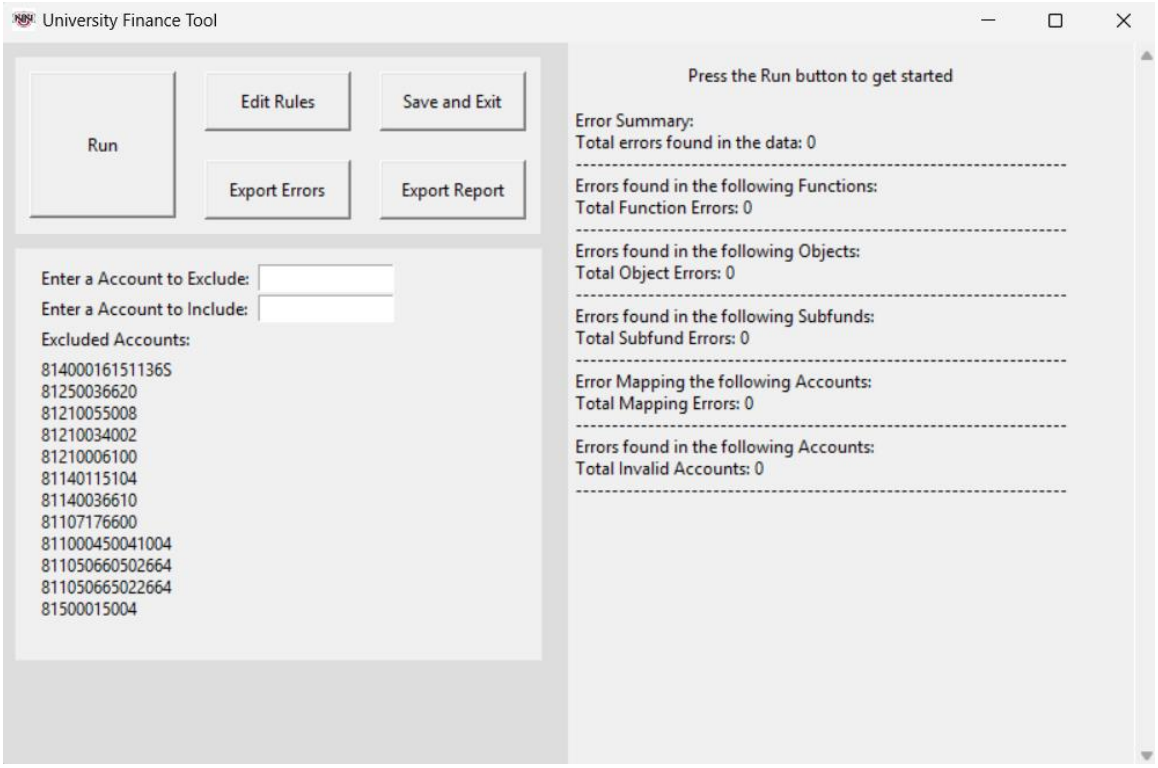
- Nothing
- Know the official "close" date for each month so I know if I pull a report on "X" date it will be updated through "X" date.

- N/A
- I'm not sure change is necessary - but having the ability to get results on a particular month's activity faster once the month has closed would be great

Appendix B: Code

Because of the sensitivity of this project, the code is available by request from the University IT department.

Appendix C: Example University Finance Tool UI



This screen shot is after a report has been generated and while it is still in memory. There are a total of 5 buttons worth noting. The big “Run” button is how you start the program processing a report. The “Edit Rules” button toggles the section below allowing it to be shown or hidden. Both of the export buttons, “Export Errors” and “Export Report,” give you the option to select a place on your computer to export the most recently generated report. Last is the “Save and Exit” button, and it will save any changes to the rules sections and then safely close the application.