Automating Contract Reports with Power Apps

CS4991 Capstone Report, 2024

Kyle Pecos
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
xxn3ap@virginia.edu

ABSTRACT

The process for creating a contract report at SimVentions had not been the most efficient, with communication issues arising among team members in the past. A proposed solution was to develop a Microsoft Power App system to store and fill out information. Users could enter their entries for projects which would be stored using a Microsoft Sharepoint database. The entries could then be viewed, edited, copied, and filtered if desired. After development was finished, these entries would automatically populate a provided Microsoft Word template, eliminating the need for them to be manually gathered and entered. Currently, the system is undergoing further development with the addition of an archiving system for older entries.

1. INTRODUCTION

Companies are always looking for new ways to improve their internal systems and increase efficiency in the workplace, lest they fall behind their competitors. This past summer I interned at SimVentions, a government contracting company located in Fredericksburg, with the intention of modernizing one of the company's internal processes. Specifically, I was tasked with working on the government contract reporting process as some inefficiencies with the previous system were very apparent. Generally, the company was given a Microsoft Word Document template and employees were to fill out and report entries detailing the work they accomplished for a particular project. After all entries were entered, the document would undergo tech-editing by a manager before being formally delivered.

This process was not without its issues as employees were often not entering their tasks on time or in the correct format. This would add extra work for the manager, delaying intended schedules. As the company was already integrated with many Microsoft-related products, my supervisor tasked me with creating a Power-Apps based solution. Power-Apps is a development tool for custom applications simplifying UI design and more intricate coding details. The tool has integrations with many other Microsoft-related tools such as Sharepoint, a management service tool for data.

2. RELATED WORKS

When researching ways to develop the Power apps system, two possibilities presented themselves, one being a Canvas App, and the other being a Model-Driven App (Yoshida, et. al., 2023). Canvas Apps allow for full customization of the user interface with some pre-made components that can be utilized if desired. It has connectors with a variety of other Microsoft services and data storage. Model-Driven apps lack flexibility in the user interface options as they only allow for preset user interface components to be utilized. Additionally, Model-Driven Apps only allow Microsoft Dataverse databases to be used to

store information. However, these apps are better for more complicated tasks that require in-depth business logic. Ultimately, it was decided to use a Canvas App due to its most robust user interface features and flexibility for database options. Additionally, the company at the time did not have a Dataverse subscription, and approval would have taken additional time and resources.

After deciding to use a Canvas App, the next step was to check whether one of the system's main intended features would be viable. As the given Microsoft Word Templates could change from contract to contract, my advisor hoped that the system would be able to accommodate a dynamic template. However, the Power Automate feature that allows for a Word Template to be populated natively expects a static template with no official documentation on how to utilize a dynamic one. After doing some additional research, I came to a blog post detailing a potential workaround to the issue I was facing (Strube, 2023). The post detailed how to parse a Microsoft Word Document's XML file and obtain tag identifiers for each field of a template. These identifiers could then be entered into the dynamic schema of the populate feature which could allow for the population of dynamic templates as long as the field names were standardized among all templates. These fields need to be manually created for each new template provided so this limitation would not be an issue.

3. PROJECT DESIGN

Before I started adding to the project, my advisor partially worked on some of the user interface components. He decided to use a color palette of different shades of blue for many of the components to match the company logo's colors. Though I was given creative control over how to design the rest of the system, I decided to continue the style that had already been set up. Copying these existing components and slightly modifying them saved me time during development so I could focus on the internal business logic of the system.

To store information about the system, we utilized SharePoint databases which could be directly connected to Power Apps. The first database created managed info on different projects from which the other tables would reference. Three tables storing entry data, meeting data, and travel data, respectively, used a foreign key to reference their associated project. In addition, another table stored the Microsoft Word Templates for the system to fill. SharePoint natively only allows 2,000 entries at once to be loaded which was an initial concern, but I successfully implemented a workaround involving iteration.

The title screen initially began with two main buttons. One would allow the user to add a new entry for a project, whereas the other one would allow the user to review existing entries. Later in development, I added a third button that allowed the user to view old, archived entries. Next to each button was a more thorough description detailing the functionality of the system.

Initially, the add entry screen was solely for adding a project entry. The user could select from a drop-down of projects they had been assigned to and create an entry detailing what they had accomplished for a particular day. After a round of beta testing, managers requested the ability to add information on meetings, travel, and future plans, which was subsequently implemented.

In the guidelines set by my advisor, the review entries screen was expected to have two modes of view, one being that of a normal user and the other that of a manager. Once all entries were entered for a project, managers could go through entries make additional edits and add

information if needed. In the case that some employees had not made any entries for a particular time period, a button could be pressed sending an automated email reminding them to do so. Once an entry had been reviewed it was marked as "tech-edited" and additionally important entries could be marked as "notable." A normal user would only be able to edit their own entries, whereas a manager would have full access to all entries for all projects they were assigned to.

Regardless of the user, all entries have the option of being filtered multiple ways including by specified project, date range, and person. Managers had additional filtering entries for non-tech-edited entries or those marked as notable. If selected, an entry can either be edited, copied, or deleted. Additionally, once the meetings, travel, and future plans request from beta testing was implemented, a separate viewing page was added to the review entry screen detailing the respective info for a particular project. As it was not anticipated for these sections to have too many entries, only an edit and delete feature was added for each.

From the filter entries screen, users also have the ability to automatically fill out a Microsoft template with the filtered entries. First, the user can hit a preview button which displays all the data to be included in the final report. If the user finds no issues with the data, they can then select a desired template to build their report with. Once the report is complete, it will send an automated email to them with the filled Microsoft Word template attached.

As the business logic for this template feature was too complicated to implement solely in PowerApps a connecting Power Automate flow had to be developed. Based on the filtering information passed in, the flow filled the template with entries, meetings, and travel data using a premium Populate a Word Document feature. However, the feature being premium initially presented an issue as only developers possessed a license to use them. Purchasing a license for every user would be much too costly for the organization. Fortunately, I was able to find a workaround by invoking the flow on my account from any user to bypass the restriction.

4. RESULTS

By the end of my internship experience, the main features of the application had largely been completed. All components of the user interface were functional with different display modes depending on the user. Database info could be accessed and edited through just the application without the need to open SharePoint directly. Entries can be filtered and automatically populated in a chosen Microsoft Word Template.

Based on feedback from project managers, additional requested features and quality-of-life improvements were successfully implemented. As some templates had sections for meetings, travel and future plans, the application had the option of filling out these sections in addition to the normal entries. Certain parts of the system were unclear to the managers at first, so I added extra clarification and directions in the app itself for future users. Last, I wrote a guide documenting how to set up a Microsoft Word Template that the system could recognize and use.

5. CONCLUSION

The application once properly rolled out to managers and teams should help expedite the contract reporting process. The Power Apps interface with the connecting SharePoint databases will allow teams to easily enter and retrieve information on their projects. Additionally, the implemented Power Automate flow to populate templates removes the need for managers to manually compile user entries as they had to do before. Through

this internship, I gained familiarity with various Microsoft services and experience developing a system primarily on my own.

6. FUTURE WORK

The next main anticipated step of the Contracting Reporting Application is the development of an archiving feature. Once a project is finished and a set amount of time has elapsed since completion, a project and all of its associated entries and data should be archived into a separate gallery screen and database. Doing this will remove unnecessary clutter in the dropdown project list and reduce database lookup times. I am currently working part-time for SimVentions and hope to have this feature fully implemented soon.

REFERENCES

Strube, S. (2023). Populate a dynamic Microsoft Word template in Power Automate flow. *There's Something About Dynamics 365.* https://2die4it.com/2023/03/27/populate-a-dynamic-microsoft-word-template-in-power-automate-flow/

Yoshida, T., Dias, S., Vivek, K., & Osborne, K. (2023). Determining which type of app to make. *Microsoft*. https://learn.microsoft.com/en-us/power-apps/guidance/planning/app-type/