Automatic Proposal Formatting Tool

Experiential Learning

CS4991 Capstone Report, 2021

Michelle Miller

Computer Science
University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
mhm9mm@virginia.com

ABSTRACT

My team and I created an automatic proposal formatting tool for a defense company that allowed compliance workers to easily format proposals correctly. I used Agile methodologies to organize the project, which we split into parts so individual teams could work on it simultaneously. The tools used in the project included: Github for version control, Python for the coding language, Jira for project management, and Zoom for communication. While putting knowledge gained from classes taken at the University of Virginia to use, the team gained many soft skills. We completed a working prototype, then turned the project over to a legacy team working for the client. The final application showed great promise in reducing the time compliance workers spent changing proposal formatting, which in turn saved money for the client. Since we were only able to finish a prototype that only included a graphical user interface and title page formatting, future work is needed to add the formatting for the body of the compliance documents.

CCS Concepts

• Software and its engineering--Software creation and management

Key Words

Agile methodologies, Python

1 Introduction

Have you ever been stuck doing a repetitive and mundane task that was simple, but took hours to do? My aim for my project was to automate a task such as this to make people's lives easier. During the summer of 2020, I was a Technical College Intern at a large defense company based in Northern Virginia. They sought a way to automatically format compliance documents to save time. Time saved would allow for an increase in productivity in the compliance office and thus would save the company money. This was an interesting problem because it allowed me to use technology to make people's jobs easier by automating a part of it.

The approach my team used was simple, we started with the prospective user's requirements and iteratively developed a solution to meet those requirements. To do this, my team used Agile methodologies which allowed us to facilitate constant communication with the client and iterative development cycles. For development, we used Python to make a working prototype.

2 Background

The company I worked for was divided into different sectors that pertained to different focuses of the business. Each of these sectors had its proprietary workflows and paperwork that allowed them to conduct business. However, when one of these large sectors was splitting into two new sectors, there was a dilemma of reformatting all the legacy documents to the new sector's format. This task fell to compliance workers, where they needed to open each document and tediously copy and reformat text. Working for a large company, they wanted to streamline every mode of work, including this compliance task. My team and I were tasked with creating a technical solution to solve this issue. Our goal was to make an easy-to-use application that was intuitive to use so that non-technical workers could use the system effortlessly.

3 Related Work

As technology has developed it has become increasingly adopted in the workplace. A study from McKinsey found that this leads to a shift in skills required for workers, including skills needed to use these new and growing technologies [4]. This shift means that compliance workers will have enough technical expertise to use my team's automatic reformatting solution to help lighten their workload.

There are currently a few companies that offer this service, one of them being DocShifter. They claimed to make word documents compliant with a specific set of rules by automating and centralizing the formatting process. Some of the rules they supported were identifying word issues, fixing styling errors, hyperlink mismatches, incorrect numbering, and table sizing [2].

While this did save time and reduced human error, it did not completely satisfy our needs of taking an already completed compliance document and reformatting the data to a new compliance document. Historically, there have been even more programs that automatically format word documents, but their capabilities were even more restricted by only being able to change simple things like margin size, font type and size, and bullet type, among other things [3]. This idea of reformatting can also be extended to web document standards, which are required to present web pages as intended. Because of broad applications of document formatting, it is hard to do at a high-quality.

Three difficulties of creating high-quality automatic document formatters have been found by three researchers from Adobe Systems Inc.: high-quality automatic formatters are hard to quantify, document layout is difficult, and designing and implementation is hard [5].

4 Project Design

Since the work I did during my internship is proprietary and some aspects are confidential, I cannot delve too deeply into the specifics of my project. I can, however, give a brief overview.

4.1 Requirements and Components

The biggest requirement was to make the automatic reformatting application intuitive and easy to use, as non-technical compliance workers were the prospective end users of the product. To ensure ease of use, my team and I decided to implement a bare bones graphical user interface, GUI. The second main requirement was that compliance workers wanted to be able to specify a compliance document they wanted to reformat and then have the application spit out another separate document with the correct formatting. This was so that the original documents could be kept for legacy reasons. To do that, we had the application digest a word document selected by the compliance worker and then select a folder to have the new formatted document to be saved in.

4.2 Tools and Skills Used

The software tools we used included Python and Github. Python was used because it is an easy programming language to spin up for easily creating prototypes. This allowed my team and me to easily change and update our application. Something nice Python feature is that it comes with a bundled GUI library, tkinter. We used this library to create a bare bones GUI for the application. Keep in mind that our goal was to only make a working prototype. Making the GUI visually appealing was not a top priority; rather, ease of use and functionality were. For proprietary reasons, I cannot show an image of the GUI, but Figure 1 shows what a base tkinter application looks like. Github was also used for version control. This allowed use to track our project and revert any changes if necessary.

I had to learn good communication and time management skills. Since this whole internship experience was online, I had to put in more effort to communicate with my college through a virtual setting. I also had to manage my time well and be flexible with others' schedules since the nature of working online is very fast-paced. To facilitate this communication, Zoom was used to meet and collaborate with others.

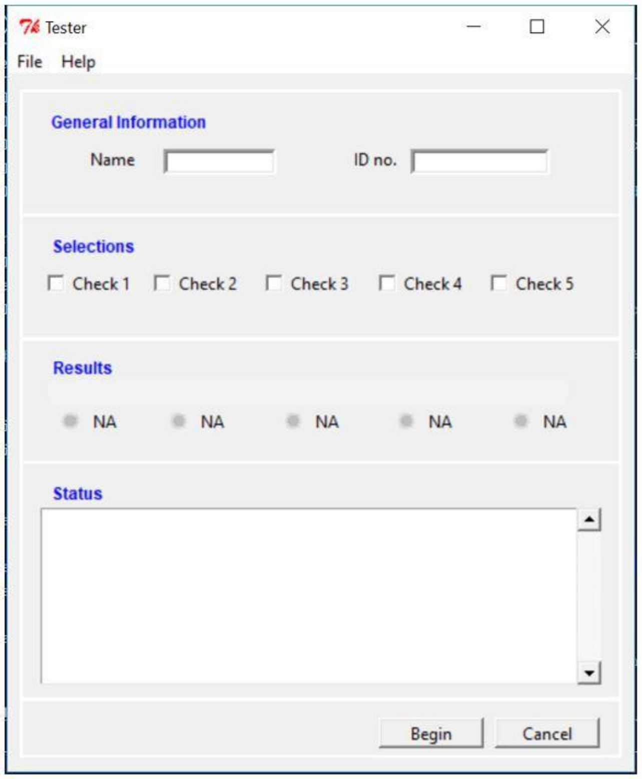

Figure 1. an example tkinter application GUI [6]

The application development was guided by Agile methodologies. This framework helped my team and me facilitate a constant cycle of requirement gathering, implementation, and user feedback. Figure 2 shows a model of Agile development with the specific steps. Jira, a project management software was used to implement this Agile development framework. The requirement gathered from talking with the client were first translated into user stories. From there, each story was ranked by difficulty. Depending on the difficulty a software engineer would take on the task and implement its solution. Once implemented, the solution was then tested for acceptance. If a bug was found, then the software engineer would have to repeat the cycle.

4.4 Challenges

The biggest challenge was keeping in contact with our client, compliance workers. It was easy for my team to find time to meet every day for scrum meetings but compliance workers had a lot on their plate and had difficulty scheduling check-in meetings. This meant that we had to conduct each meeting with our client effectively and efficiently to gather all of the requirements and changes they needed.

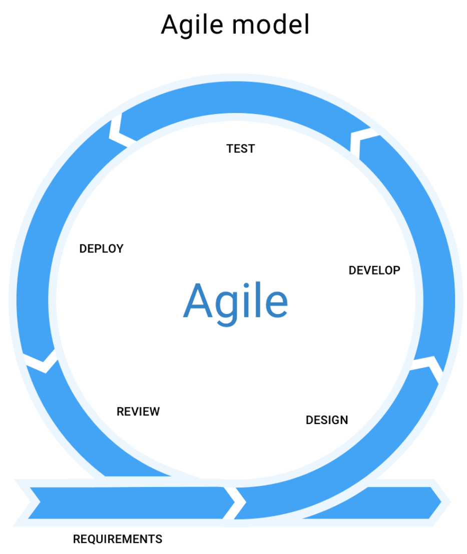

Figure 2. An overview of Agile methodologies [1]

5 Results

By the end of the summer, we found the feasibility of such an application developed in-house. It was exciting because the company I worked for saw potential in it and this project got passed onto a legacy team that continued the project. What we were able to accomplish was the original compliance document digesting and reformatting the title page, headers, and general body of the document. We were also able to save the newly-formatted document into a new file in a folder specified by the end-user. Since the end goal was just to complete a working prototype, our solution is yet to be used throughout the compliance office.

6 Conclusions

Although our technical solution was only a working prototype, it had the potential to save compliance workers a significant amount of time by taking away some of the repetitive and mindless tasks they had to perform and, increasing their productivity. By extension, this increase in productivity would have saved the company I worked for money by not having extraneous billing hours from compliance workers. This productivity was acknowledged by the senior management, who allowed a legacy team to pick up the project after my summer internship ended.

7 Future Work

A lot of work must be done to make a final application that can be widely distributed to compliance workers. First, the GUI would need to be updated to be visually appealing and more polished than the simple tkinter one that was used in the prototype. Second, the formatting rules must be expanded. We were only able to implement a subset of the required rules.

8 UVA Program Evaluation

The concepts I used in this internship all came from the coursework of CS 3240, Advanced Software Development. This class was very similar to my internship experience where you were thrown onto a random team and had an extended amount of time to complete a project. Through the experience gained from that class, I learned how to work with strangers in a team to accomplish a goal, how to use Github, and, above all, how to use Agile methodologies. Although the introduction to Agile methodologies significantly helped me in my internship, a student would benefit from a more structured sprint cycle. When I took CS 3240, the sprint cycle was loose and did not cover all the specific phases used in actual practice, such as a retrospective. Students would also benefit from a structured mock client interview. Within CS 3240, although there is a requirements manager, a lot of the requirements are already assigned. I believe there was an interview portion of the requirements, however, those interviews were most likely done with a student that does not yet know how to express exactly what they need. Since the class is already split into cohorts with teaching assistant leads, this one-time mock interview could be easily implemented into the program. I would also like to see more group work throughout the computer science curriculum. CS 3240 was my first experience with this group work structure, which closely aligns with the way software development is conducted in the real world. The more exposure that students have to this group structure, the better they will succeed in real-world development after college.

REFERENCES

[1] Anna Dziuba. Agile Software Development Lifecycle Phases explained. Retrieved October 26, 2021 from https://relevant.software/blog/agile-software-development-lifecycle-phases-explained/
[2] Anon. 2021. Auto check and FIX word documents for compliance. (July 2021). Retrieved October 26, 2021 from https://www.docshifter.com/solutions/word-formatting-fixer/
[3] Anon. Auto Formatting in Word 2010. Retrieved October 26, 2021 from https://www.tutorialspoint.com/word/word_auto_formatting.htm
[4] Jacques Bughin, Eric Hazan, Susan Lund, Peter Dahlstrom, Anna Wiesinger, and Amresh Subramaniam. 2018. Skill shift: Automation and the future of the workforce. (January 2018). Retrieved October 26, 2021 from

https://www.mckinsey.com/featured-insights/future-of-work/skill-shift-automation-and-the-future-of-the-workforce

[5] Nathan Hurst, Wilmot Li, and Kim Marriott. 2009. Review of Automatic Document Formatting. Proceedings of the 9th ACM symposium on Document engineering - DocEng '09 (September 2009). DOI:http://dx.doi.org/10.1145/1600193.1600217

[6] Sanchit Gupta. 2019. Executable GUI with python. (June 2019). Retrieved October 26, 2021 from https://medium.com/lifeandtech/executable-gui-with-python-fc79562a5558