A&M PowerPoint Tool Outline

Project Goals

I am tasked with creating a PowerPoint tool intended to aid and speed up the process of putting together slide decks when applying for RFPs and other jobs with clients. The tool is aimed to make people's lives easier by providing them the slides and materials needed upfront, in order to avoid having them jump through hoops.

Solution Overview

To accomplish this goal, two tools will be created. One will be for initial creation of a deck, and another will be to add slides to a pre-existing deck. Ideally these two tools can be used in tandem to create a deck from scratch, or individually as well.

Tool 1:

The first tool will be an interface that a user can interact with to dictate what kind of slides they want in their presentation. We can give various options, and I will work with the team and discuss with the users what types of slides they like to see in all of their decks, and figure out where the variation is. This will help autogenerate a template that is specific to the company or job we are applying for.

Tool 2:

Tool 2 will be a gallery of slides from all the slide decks from the teams. There will be a search bar and a way to narrow down the results to be more specific. I will work with the team in the future to see how they would like to find their results. After typing in your term, all relevant slides from all decks will appear underneath the bar. Ideally, this will be ordered by relevance (that will be a later addition). The user will be able to (hopefully unsure on this part) see the image of the slide, and click it. When the slide is clicked, it will be inserted into the presentation. Another later feature will be to have the slide that is inserted have the previous information stored from tool 1 (such as title) automatically inserted into the slide.

Resources to be used

There is two main libraries that I will take advantage of.

Library 1:

First is the PowerPoint task pane add-in API. Microsoft has a system to create a plugin for PowerPoint that will set up on the right of the toolbar. This will definitely be used for tool 2, and hopefully for tool 1. The biggest problem with this API is there is incredibly little documentation on what the software can be used for. On the official Microsoft website there is only the example of adding elements to a slide, and after looking through the web I found very little information about what else can be done. Despite this, it seems that it is very easy to develop a UI with this tool (HTML), and the backend to this HTML is Node.JS which I know. I haven't actually gotten my hands on the API yet as it requires that you develop in Visual Studio Code, which I need to setup. A lot of the information on the website is for the general office suit, not PowerPoint in particular, so it could take me a very long time to understand how to fully utilize this library.

By going this route of having it in the PowerPoint ribbon (in my opinion the correct path), it means I need to create this project as a web application. That means that I will need to deploy and host the front end on a web server (which will cost money) in order for the team to use it. This also means that we will need to create a login system (I am probably going to use OAuth and pair it with Microsoft, so the team can simply use their Alvarez logins).

Other resources will be needed by going this route. We can use a SQL database, but it may be very difficult to use the Dallas server. The web application will need to be able to login and retrieve the data from the SQL database, and I highly doubt that it will be able to. I recommend that we use a NoSQL MongoDB server that is hosted by Mongo for free, as it is very secure, cheap, and I have used it before. We also could use a MySQL server, but again this will all come down to how we host this web application.

If we are hosting the application, it is very likely that we will also have to store any input decks that we want to use on the hosted environment as well, but I could be wrong and maybe I can tap into Box's API in python to access the decks from anywhere.

I also am unsure on how difficult it will be to deploy the application. Based on the research that I did do it doesn't seem too hard, but also deploying things is never simple. There also seems to be a way that we can deploy it exclusively for A&M employees which would be very ideal, but to do this we will probably have to work very closely with IT.

Library 2:

The second, and way more powerful library is the Python-PPTX library. This is how I will accomplish tool 1, as you are able to manipulate a lot of elements on a slide. I believe it is powerful enough to scan all of the slides in the deck to also create the catalog needed for tool 2. I will try to create a backend that will revolve around this library, but in order to do this I will need to figure out how to call the script from Node (I know you can, I am just not 100% sure how that will work on a web server). I would use this library to do any large scale operations on the decks.

How it would be structured

Tool 1:

Tool 1 will have a HTML front end in the PowerPoint task bar, that when clicked will have a bunch of inputs users can choose. Then from there it will launch a python script (hopefully) that will scan through a master slide deck that we create with anything we may need to start up a deck. It will create a new PowerPoint document, and insert the slides (hopefully) we want into this new deck. Then it will open this new deck.

Tool 2:

Tool 2 will also have a HTML front end in the PowerPoint task bar. It will have a search bar that users can search terms to find the right slide.

In the backend, there will be a python script that will go through all of the slides in a folder, and insert all the data from the slides into a well-organized database.

Then the tools backend will connect to this database, and search through it to find relevant slides. Then by a combination of stored files, python, and maybe node we will present the correct information to the front end.

This front end maybe will need to a little more complex, with maybe jQuery elements, (I really don't want to have to use react or a front-end framework), to create an HTML element for each of the search results. The goal will be to mainly use JavaScript, as PHP would be too difficult. We will present these new elements below the search bar. One a user clicks these options, I will (hopefully) use Node to insert the slide into the deck. This will come down to how we can store the information in a slide. Worst case scenario we will have to store the information of each element on a slide and how it is organized and just inserting these elements into an empty slide, but if this is the case the project will take much longer

to complete. I am hoping there is an easy way to move and arrange whole slides, but I am unsure as of now on how to transport a whole slide from one document to another open document, but research and testing will be needed. The amount of flexibility that is available for this transportation will determine how complex the database will need to be.

Summary of Potential Issues

I think this is a very important thing to discuss before we move forward. If we want this project to be something that you can click in the toolbar, it will require a lot more steps. I think this is the best way to proceed, as it makes for the most user friendly experience, but it will cost money, take a lot more time, and we will have to consider security risks. I think the only alternative would be a .exe file that a user would have to launch, but even that could only accomplish tool 1, not tool 2, or we would have to rethink the second tool.

I can develop both tools, but in order for the rest of the teams to use them we would have to deploy it. There is no way around it like with the Opportunity Hunter tool, we cannot just host it off a Magnus tool. I don't need the servers and databases to develop (well not really), but I rather not spend a lot of time creating something that we ultimately can't use.

I think the security risks are relatively low, but again I am not the most experienced at developing applications that can be used at a corporate level, as what I have been has been mainly school projects with not sensitive contents. If the slide decks are something that we couldn't afford to lose, then maybe this route is not the way to go, as I cannot promise that an application that I will have to code from the ground up will be incredibly secure. There are measures that I can take to keep the common person out, but an experienced hacker would not be someone I could stop.

It mainly will come down to how we would host this web application, because doing so locally would require us to download all the files and deploy it on everyone's computers, which would be widely inefficient, especially as we would have to repeat the process every time there was a change.

I will also need to figure out how easy it would be to call a Python script to act on the pre-existing slide decks from the web application, as that functionality would be key.

Timeline

At the rate of working around 10 hours a week and accommodating for weeks that I physically cannot fit any work in, I think the rough timeline of the project would look like so:

1. Week 1: More research on what each of the libraries can do and setting up VS Code.
2. Week 2: Begin working with Python PPTX to see how we can manipulate slides.
3. Week 3: Create a mock UI somehow in Python (maybe the command console) that users can interact with and give the tool information.
4. Week 4: Use this information to manipulate a slide deck to create another. Have a working prototype of Tool 1.
5. Week 5: Figure out a way to move the front end to the Office Add-in setup, and have a working prototype that can be used in PowerPoint.
6. Week 6-7: Work on the Python script needed to scrape through decks in a folder and catalog all of the slides. Make sure to do this in great detail so we can scale it.
7. Week 8: Setup the database for all of the scraped information from the slides
8. Week 9-11: Create a backend that can access the database, present previews of the slides to a front-end, and insert the slide into a deck.
9. Week 12-13: Create a front end in PowerPoint for tool 2, and connect it to the backend.
10. Week 14-15: Figure out a way to securely deploy the project.
11. Week 16: Work with IT to ensure the tool can be used by the team.

My estimates could be widely off, but I feel that my overestimates and underestimates will result in the same result. Due to my limited hours, it will probably take around 4 months to fully accomplish the goals set here.