Team Name: The Libarbarians

For this milestone, we will provide what key tools, technologies, and the process model used for this course project. Below shows our list of items that will be utilized throughout the process of our Librarian Assistant project.

Tools:

- Microsoft Visual Studio Community
    - We will be using Microsoft Visual Studio Community as our IDE because we are all familiar with using the software through our other courses. It's a useful and intuitive program for coding in C++, and using a software we all have a baseline familiarity with will improve efficiency.
- Google Docs
    - Google Docs provides us with an easy way to share master documents that can be edited and shared in real time. Multiple people can concurrently work on a single document without issue and remotely. We will be using Google Docs to document our code and plan ahead during development. We decided to use this instead of Word because of how easy it is to access for all members.
- GitHub
    - Github is an industry standard and works best for version control, workflow with collaboration, and documenting bugs. It allows us to utilize branching code to offer new ideas or changes to the main branch of our project, as well as allowing our group to collaborate on the same coding files. Additionally, allows us to access all changes we have made throughout the process and track all versions of code.
- Discord
    - We will be using Discord as our main communication application, allowing us to post and answer questions effectively about upcoming deadlines, milestones or clarifications. Discord provides a familiar and definitive

platform for team members to communicate through voice calls, screen sharing or instant messaging.

- We chose Discord over other options (such as a text message group chat) because using a third party application lets us access the chat on computer or smartphone. Additionally, the Discord chat program is very effective at sharing bits of code which will be useful as we move forward in production.

Technologies:

- C++
  - We chose to code the majority of this project in C++ because all group members are familiar with C++ from CPT_S 122 and 223. C++ provides many useful object oriented programming structures while maintaining a high level of performance. Additionally, Visual Studio Community provides a platform for Windows desktop applications using C++, which we plan to utilize.
  - Memory management in C++ gives us full control of when memory is freed, as opposed to in Java where it's handled by the garbage collector.

- Windows API (Win32 API)
  - We will be using Windows API to create the desktop application of our project. We chose this method because Windows API has been in existence since the 1980's, meaning there are extensive resources for using this framework which we will need as no one in our group has proficient familiarity with creating desktop applications.
  - While Windows API is an older framework, during our research we found strong arguments for using and learning this framework, one of them being it is still used commonly today underneath other frameworks such as MFC and .NET.

Process Model:

- Weekly team meetings on Discord Wednesday nights at 6:00pm.
    - Our team finds communication of the utmost priority and consistent meetings will ensure we are working as cohesively as possible. We will be meeting in person as the project starts to develop more into a production stage, but during the planning stage we will be utilizing Discord's screen share feature to discuss timelines, models and milestones.
- RAD Model (Rapid Application Development)
    - The reason we chose the RAD model is because it allows us to complete individual aspects of the code extremely quickly and coherently. There won't be any miscommunications or difficulty understanding what certain sections of code do as it'll all be done by one member. Because the coding time is so efficient, that provides lots of time at the end for combining all the code together, which we understand will be a large portion of the production timeline. Because the RAD model favors speed, we can afford to spend the extra time at the end to edit our program as a team.
    - Each individual member will be in charge of their own small segment of the code. This way, each member can design an internally consistent model with no "growing pains" arising from two members stitching similar code together. After all the models are independently developed, they can be retrofitted together at the end
    - This also helps with communication from the customer, since if they have a problem the only person that needs to be alerted is the one in charge of that specific section. Thus there will not be miscommunications over solutions between team members.
    - We believe that having each member focus more of certain parts will increase our attention to detail and make for more thorough code as opposed to having all members be responsible for all components of the project.

- Communication and modeling at the beginning of the project will be essential, as once the roles are locked in, code won't necessarily be shared between members. We will talk about ideas amongst ourselves, but we will not be sharing hard code. Because of that, we need to pay careful attention to make sure that each team member is on the same page with their vision of the final product.