Milestone 1

Team Name: PM-Tool

Members:

Christian Rouhana

Daniel Lee

Liyuan Wang

Shinichiro Yamagami

Paiyi Wang

Key Tools:

Microsoft Visual Studio

As this is a collaborative project, we want to use Microsoft Visual studio for its strong integration with version control tools like GitHub. Visual Studio also has great GUI tools for compilation and debugging, as well as easy drag and drop features for creating GUI based applications without heavy use of Microsoft's markup language.

GitHub

GitHub is the most popular version control tool available. Its online environment is clean and easy to use, and it allows our team to quickly collaborate on the code/documentation base without stepping on each other's toes. This also allows the TA and professor to check our work, and GitHub is an easy way to distribute our software once it is complete, thanks to its "Download as ZIP" button for any repository.

Google Drive

Maintaining documentation will involve the entire team to various extents. To do this, we need to have an easy way to work on the same document fluidly, without having to deal with repositories or sharing ".docx" files. Google Drive has all functions that applications like Office have, including being able to easily export our documentation as a PDF, making sure the design of our documentation is preserved for the end user.

Microsoft Windows

All of our development will take place on Windows. The reason being, Windows is the most popular OS in the world. Our whole team knows how to use Windows, so there won't be a need to get anyone up to speed on a UNIX environment for example. Also, developing for Windows allows us to easily make our application optimized for Windows. Our end user will likely use Windows, so that is the most important platform for us to target.

Undecided UML Diagram Tool

While we have not settled on a UML tool yet, we will heavily integrate UML diagram tools in our project. UML modeling will allow us to have a cohesive idea of the structure of our code as the code base gets larger and larger. It also allows any member to know how something works despite their involvement. For example, if one member is mostly focused on documentation, they can refer to the UML diagram to better understand how the program works.

Discord

Discord will be our main communication tools for the project. We will plan any future task and discussion about the process in this digital distribution. Discord has a great function to share ideas such as voice call voice calls, video calls, text messaging, media and files.

Technologies:

UWP Applications

The output of this project will be a Universal Windows Platform (UWP) application. This is because most of our target user base will be on Windows. This brings a host of benefits, like familiar user interface as well as the ability for us to target one platform and optimize the performance.

C# & .NET

Java is a common ground for our team, and as C# has a similar syntax, we are going to use C# to take advantage of many of Microsoft's .NET libraries as well as WinForms to create our tool. This is ideal for our team as we won't have to track down a third party Java library to create a strong GUI for our application. We also will be able to get to grips with the C# syntax and quirks very quickly.

WinForms and XAML

These tools from the .NET library deserve specific discussion for our project. They will be the core of how we design the GUI for our application. While Visual Studio allows for easy drag and drop GUI design, we will still need to interact with

the “.xaml” files as well to refine our application. These tools will allow us to easily make a recognizable and effective GUI that Windows users will be able to easily use.

Process model:

We would adapt the evolutionary process model.

- The main reason is that we already know what we are going to build and what functions are necessary for it but we have not discussed what other features we are going to add. Those features will be added at any time we feel them important or necessary.

- This is a similar reason above, but we first want to build something working and test it. And then, we communicate with each other on what features to add, plan, build, and test them repeatedly. This process model is adaptable and lets us build plans and implement them step by step.

- In addition, using evolutionary process model allows us to get fast and early feedback after each working part or testing, which can be very helpful as we discuss the other features in the future to adapt any changes if needed, and working toward the goal.

- One of the disadvantages by adapting this process model is hard to develop software concurrently. But our group is small and the time is limited. So it is not necessary to worry about the concurrency for the project.