Ethan Villalovoz, Ryan Aloof, Roy Zabetski, Bryan Frederickson, Nicholas Santos

Project Milestone 1 Sunday, January 29th, 2023

Team Name: Five Guys One Chair 🪑

For this milestone, the team will provide what key tools, technologies, and process models used for this course project. Below, are all 3 of those, divided into different sections to provide a structured layout of the plan as of now:

Key Tools 🪑 :

- Github + Git:
    - We will use Github and git to collaborate on the project and work on its different versions and features through branches. GitHub will also let the TAs stay updated on the project's current state and follow along if any questions arise through the development cycle. The Issues Section in Github will log tasks and troubleshooting, and the projects section will assign and keep track of each team member's tasks respective to the issues section, providing a clear overview of our development.
- Microsoft Visual Studio Community 2022:
    - We are using Visual Studio Community 2022 as our Integrated Development Environment (IDE) for writing the application's source code with the experiences from using it in other classes. With its capabilities to design an graphical user interface (GUI) for user interaction with ease with powerful features and the availability of a comprehensive set of tools and libraries. The language of choice for developing the application is C#, complemented by XAML, providing an efficient and effective development environment.

- Google Docs:
    - We will use this platform's online document editor for its live-time documentation updates when collaborating simultaneously with the team. Google Docs accessibility through desktop and mobile apps allows us easy access whenever it is required to view and update changes to our documentation and tasks. Compared to other online document editors, it is more user-friendly to allow other external team members (the client) to efficiently contribute to the project planning. With the vast tools integrated into the software, we can customize and illustrate our ideas expressively.

Technologies 🪑 :

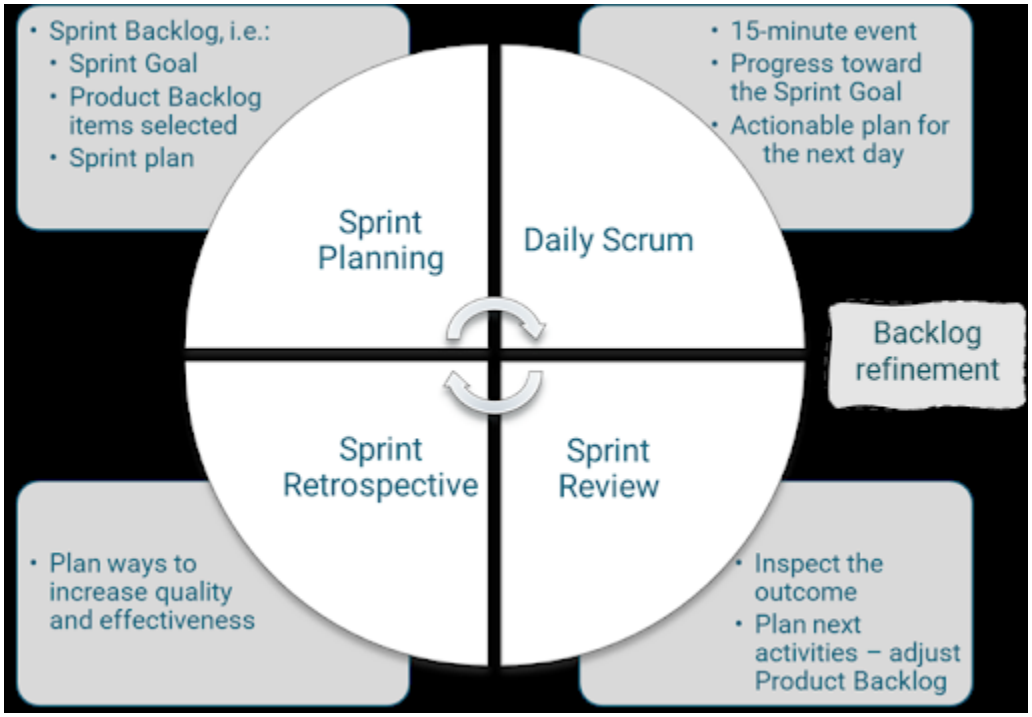
- WinUI:
    - WinUI is a well-documented and easy-to-learn user interface framework that targets windows development. This would be an excellent framework to create a prototype for our program. This is because we have a C++ background, so it will be easier for us to learn the C# framework that WinUI utilizes since C# is a C-like language. Also, we would be creating an application with WinUI, so we would not have to handle creating a domain for a web app, which may bring up unexpected issues. Finally, WinUI strikes a good balance in platform targeting since we could easily migrate to cross-platform tech that relies on WinUI, like React-Native for Windows, UNO, or .NET MAUI.
- Offline Database (SQLite):
    - Since this is a prototype, we'd like to keep this project offline and avoid the costs and headaches of deploying a live database solution. We feel using a technology like SQLite will allow us to achieve this goal. In addition, we plan to architect our database code to be modular, so in the future, we can swap out an offline database for a live database. We do this to keep development fast and straightforward while providing the scaffolding required for the live database functionality we may seek.
- React (Not Using):
    - We initially considered building a web app using the React framework but decided against this for several reasons. First, React is more challenging for our team because we have a C++ background. React is built on top of JavaScript, and while it is more practical long term, as a prototype, we feel this is not suitable due to the extensive learning curve that will come with learning JavaScript, React, and other required WebAPIs/Frameworks/Environments like Node.js, Firebase, and Azure.

- ChatGPT:
    - Chat GPT will be used to support our task creation feature. This is because ChatGPT exposes a public API that we can utilize. While using ChatGPT is a feature of our app, it is not the core of its functionality. If ChatGPT were to go down, our app would still function as a regular task app. Currently, we plan to use the text-davinci-003 language model that OpenAI exposes in the ChatGPT public API.

- Autofac (Dependency Injection/IoC Container):
    - We aim to use a Dependency Injection and Inversion of Control framework to make the application testable and extensible. The first one that comes to mind is Autofac since it is a mature framework and has shown to work very well with C#. This will allow us to have a top-down kind of dependency graph, which will give us flexibility in making changes to our architecture in the future via the use of interfaces. Dependency injection allows us to quickly test our code since dependencies can be swapped out for test versions instead of requiring us to mock accurate data.

Process model 🪑 :

The process model used here will be the **Agile Scrum/Sprint Model**. Here is an illustration example of the model structure here



- **Pros of the Agile (Scrum/Sprint) Model:**
    - Agile Methodology, in general, provides an easy way for team members to work on the project's features simultaneously. Because of that, we get quicker feedback, adaptation, and development benefits. There can also be better developer engagement and collaboration because everyone is assigned a task.
    - A second advantage of this process model is that the software production retains high quality through integrated testing throughout the development process. Benefits from this methodology allow us to identify and resolve possible defects and collaborate better with the team to improve the quality of the final product. Its proper emphasis is the extensive testing throughout the project timeline.
    - The Agile scrum/sprint approach also benefits the client because of the continuous feedback that occurs during this process. This feedback can help

developers continuously improve the software to make it higher quality and better fit the client's needs.

- A final benefit of this process model for our project is the advantages of our C# testing framework through its compatibility through streamlining the testing process. We can achieve increased collaboration in development through testing to ensure quality deployment, which allows us to achieve this quicker. Many tools integrated into the C# testing framework allowed us to utilize and adapt to any situation, such as through Microsoft Visual Studio Community 2022.

- **Cons of the Agile Model:**
  - Due to Agile methodology being very iterative and fast-paced, there are risks of miscommunication between team members and the client. This can affect the delivery and quality of the project as a result. While this is a notable disadvantage, it can be avoided. The team must have a robust system to track everyone's tasks and clear communication lines.
  - Another disadvantage of this process model, despite it not being apparent, is the flexibility it has during the development process. Yes, this is a great thing to be able to pivot in different directions based on deadlines; however, these rapid changes can lead to poor deliverables and missing deadlines for the clients when presenting. The Agile model focuses on planning, producing, and documenting software quickly and iteratively. Although these are significant advantages of this model, the quality and the uncertainty of the project timeline are at risk as there would be a lack of uncertainty of the client's expectations for each deliverable. With all of these complications at hand, this can seep into the team's dynamics and lead to frustrations amongst each team member, which will hurt the success of the software project deployment. To ensure this does not occur, it is required to have a high frequency of regulating checking with each team member and the client to ensure all of the expectations are in line.
  - Agile methodology, while it is robust, can, at times, lack structure, discipline, and documentation. Newer team members may take longer than usual to get

onboarded due to the fast and iterative pacing of the scrum/sprint method, and the limited documentation can make it difficult to track the team member's progress. Teams can mitigate these risks by compensating rapid delivery for flexibility within the planning and documentation process.

We think these is the appropriate tools, technologies, and model necessary to get the job done as we envisioned it (for now 🪑).