Developing a game is no easy feat by any means. The creation of a game takes multiple disciplines, skills, and dedication to make something comprehensive and interesting to play. In all respects games are an art that can be interacted with, a story you can delve into, and a song that is enjoyed many times over. This creates a tall order for the developers of a game to create high quality games in a magnitude of shapes and sizes, and as such a proper development process can help to bring out the best in the development team while staying cohesive to a set idea.

When it came to planning our project I felt it could have gone better and that I personally should have done it differently. I had taken an idea that I had on the backburner and proposed it to the team as a baseline idea to get the conversation going, however this game idea was drastically different from what the team wanted to develop. We had decided to do a top down shooter while my idea was built around a first person shooter, this caused a great deal of my original idea to be unusable and I feel generally altered the brainstorming process negatively. I believe this is because of the order in which I pushed the conversation for the brainstorm. Instead of shoveling a premade idea onto the table I should have first looked to the team for ideas revolving around the top down concept, and cherry-picked ideas from my first person shooter to add to the conversation. In retrospect I feel this would have made a more cohesive design than what the team accomplished. I think I was too eager to get the ball rolling at that time and should have taken more consideration of the thoughts of my teammates.

Once we had the amalgamation of a game idea we decided on a development schedule which was rough around the edges. Additional detail should have been added to the development plan to give us a more granular checklist for what to accomplish each week of the development cycle and for each milestone. However, the lack of a good foundation may have made it ultimately difficult to create a to-do or checklist for what to accomplish each week. Despite this it did temper our scope creep oddly enough by looking at the overall amount of time we had for development and what we individually felt was feasible at that time. In hindsight I had misinterpreted the course schedule and thus had to alter the last week's development timeline but seeing as we did enough to meet other criteria or could alter our plan to meet another point in the criteria we did not lose much. In the end I felt this last minute change was for the better and something that could have only happened in such a scenario. At the end of the day though we did have to alter the scope of the project due to poor planning at the beginning of the project's cycle. On the other hand, for the planned features we did have we gave ourselves ample time to complete since we knew we had a limited window for working on the project. Overall we planned for one general feature each week (powerups, weapons, enemies) and built upon any framework that had been established later when testing.

With regards to the overall development process I can say that I personally enjoyed it. I enjoy coming up with a task and completing it, but I always feel that I can do better than what I did if given more time or knowledge on the subject matter. With regards to my team I can not say anything for certain outside of what I have observed over the past few weeks. Communication between the three of us was good, not great as it could have been better, but we kept in touch where it mattered. Every week a message would be sent out on assignments and to make sure that everyone knew what they were going to be doing. A voice call was held towards the end of the week to check up on what we had done and assist one another on issue points. If a team member had additional issues after the voice call we would occasionally have another voice call on the weekends to resolve any issues. Overall our communication was lax but effective enough for the task we had with my only complaints being that I mainly led the direction of the discussions. I felt that if I did not speak about something that nothing would be discussed or there would be a severe lack of communication. Perhaps this is the job of a project manager to some degree, or perhaps we ultimately did not work well as a team - I can not say for certain. But overall this communication method of a weekly message and call led to a more productive development process I feel as it allowed us to be open about our issues and ideas, knowing that if we sent a message we would get a reply, and if absolutely necessary we could get on a call to help one another out. With how different our non-school life schedules were this also helped with reducing stress amongst the team knowing that they could work when they were available to do so and still keep in touch beyond the scheduled call in at the end of the week.

Having a lack of proper or otherwise fleshed out documentation for the project left the scope open for interpretation. While I feel that the team did not stray too far from what little scope we did have, mainly due to time constrictions, I do feel it could have been fleshed out more than it had been. Because of this I feel it is hard to say when and where the programming did not align with the documentation as we did not have much to align to other than “make this thing.” The only things I can think to mention is where there is a lack of implementation on the part of the guards weapons. Initially the guards were to have a variety of weapons, two of which were added, and any others removed due to time constraints. Despite this the guards are only ever seen using one weapon, the assault rifle, of which the guards ability to fire projectiles is not attached to. Meaning that the guard could drop its weapon and it could keep firing, or any changes done to the assault rifle such as fire rate or damage would have no effect on the guard. The firing ability is detached from the weapon itself. I had initially planned to have the guards attack based on their weapon type and while the guards dropping their weapons is set up in a way that they can drop any weapon they can not handle any weapon and are just mimicking using an assault rifle.

When it came down to testing the features we had implemented it was left mainly to whoever had designed and implemented the feature. For me personally I have the habit of testing whatever I add immediately to see if the feature works as I intend it to, and if not then I adjust values in small cases or rework the code in bigger cases. Then once complete I share it with the team to get feedback based on what they can find in their own testing, with usual bugs appearing with Bitbucket LFS not copying and storing necessary files. One such example is with the guards model holding the assault rifle. A socket is used to attach the assault rifle to a given position on the model with the idea that each weapon type would have its own socket. However when pulled from Bitbucket the socket would not be saved to anyone but the creator. This caused an emergency meeting to figure out what the issue was, consisting of the other members screen sharing the code while I overlooked it and suggested changes. In this instance I had all the changes memorized and was able to quickly resolve the issue. Other instances consisted of Bitbucket and its LFS not transferring data properly which also consisted of similar resolution methods as the guard or otherwise was other team members looking over whatever changelog I posted to ensure everything was present. One situation of this that was recent was the second stage having missing door locks that caused null reference errors. Having them push and I pull gave me the issue and I was able to resolve it easily. Beyond Bitbucket tweaking out and not working as desired most of the bugs were resolved before reaching the repo, with only minor changes being made to the various features after testing such as weapon damage and AI behaviors. If there is to be a next time I would still do minor bug testing on my own but having documentation made for how the bug was resolved I have heard is typically a good idea, so that in the event the bug reappears the archive of bug fixes can be consulted to see if it can be fixed again in the same way. This would also allow other team members to learn from my mistakes and vice-versa improving the time it takes to resolve bugs. In addition I would also urge team members to keep track of any changes they made so that when we test their changes we know what we need to focus our testing efforts on.

As a team we did not use many tools overall with only minor issues appearing with Bitbucket and occasionally on my end with Unreal Engine. As previously mentioned Bitbucket would occasionally not store or update files that had changed correctly leading to errors when pulling someone else's work, and leading us down the rabbit hole of figuring out if it is us or Bitbucket that is the issue. For me Unreal Engine also gave me some minor headaches but I feel this is more due to my lack of experience with the game engine and my experience using the Unity engine. Blender and the exportation of models from sketchfab, a 3D model distribution website and converting the files from what was on sketchfab to something usable in Unreal took some getting used to. Sometimes the models would import into Unreal without textures which would lead me to a wild goose chase of figuring out why that happened. Luckily I could just rip the texture file provided and drop it into Unreal and use it that way, something I did not realize until late in the development cycle.

I can not say for certain what programs the rest of the team found useful or not outside of some trouble with Discord not recognizing one person's headset so he was silent for a small portion of a meeting. But I can say that our collective use of the Unreal Engine and the way that the engine was developed helped us in the quick iteration of ideas, something I can not say the same about in Unity. This falls directly to the visual scripting components or Blueprints that Unreal utilizes. While we did set up the project to handle C++ scripts we mainly stuck with Blueprints as that was what everyone was used to and the limited time that we had pushed us in that direction anyways. Using another engine would have slowed us down and jeopardized whatever quality we had due to lack of experience with said engine, and the lack of a visual scripting component as well would have altered the final outcome of the project.

Beating the dead horse, our lack of a fleshed out project document did not give us any insight on what programs to use outside of the game engine, communication tools, and repository services. We essentially looked for free tools to use as we needed them, for models we used Sketchfab, for textures we used Quixel Bridge, for sounds we used freesound.org, and Blender was used to convert models from Sketchfab into usable Unreal models. Outside of these tools we did not add or use anything else as far as I am aware at least, with the team's general consensus being that if a tool does not work for us we do not use it - with the majority of our disappointment being around Bitbucket as previously mentioned.

Overall this project was a whirlwind of complications and learning experiences that had to be overcome in one way or another. For the future I now know to have as much detail as I possibly can think of when creating a project document before even creating the project. This I feel was a lesson I have already learned as I am documenting, archiving, and adding to different game ideas on the backburner as time goes on but perhaps I am lacking the formal or professional touch for an actual project document. Next would be for if I become a de facto project leader or manager again and that is to let the team lead and work as a director to guide the team in a direction rather than a conductor or leader forcing the team in one direction or another. Another lesson would be to get to know the team better, not knowing the strengths and weaknesses of the others I feel led to a worsened timeline and underestimations when it came to what we could or couldn’t accomplish for the project. I had initially planned in my head to have some sort of “exercise” to get to know my teammates better but once we got into that first meeting that all fell away pretty quickly. Lastly is to look more at the set timeline and to use the project design to set reasonable due dates and milestones. While setting due dates was not too much of an issue I feel that having more detailed project documents would alter due dates to some degree that should be considered.