I’m going to make my text blue! - Trevor

Red - Jon

Hello, I am magenta - Kyle

// I forgot the exact distinction we made between ‘player’, ‘user’, and ‘avatar’ in our meeting. It would be helpful if someone could write it down in this document. Thank you.

**The player** refers to the people who will be actually playing our game, aka the CS students.

**The user** refers to the theoretical users of the software in the developed scenario.

**The avatar** refers to the character that the player controls in the game itself (Although it seems like we might not have a representation of that avatar with the interactive text approach to the game).

**Scenario Development**

**Exposition**

**What is the setting for the exposition - this is the initial scene where the viewer gets a sense of the context ( is this an office, a house, a school..etc..)?**

The player will start off in an office environment where they will be introduced to the different elements of the scenario (coworkers, boss, what type of work they are doing, etc.). I think the office environment should be modern with lots of open space as opposed to traditional cubes and offices. The in-game office setting is necessary to simulate a real-life office environment where the person playing the game might work some day. Having a more modern office will help show the player that the scenario is taking place in the present day. This is in contrast to having a more standard setting with cubes and offices that has been used for many years (so it is harder to pinpoint the exact year the setting is taking place). Along with the office setting, the people should also be wearing business casual / casual Friday type of clothes as is common in more “modern” tech offices these days as opposed to wearing a full suit.

I think that ideally we would try to present like “a day in the life of a software engineer” or “a week in the life of a software engineer”. For the initial scene, I would suggest starting at the home of the avatar in an early morning, and heading off to work. Perhaps an interactive element could be, what would you like to bring to work in your backpack and it might have some slight variance of outcome on the story.

I do like Kyle’s idea of starting it off at home, but I think the context we need to make sure to touch upon in the exposition is that of the avatar’s company. What if we were to start it off for an advertisement of the insulin pump? (Sort of something [like this](https://www.youtube.com/watch?v=5b10pffnxVI) - this isn’t an exact match of what I’m thinking of, as the device in that ad is for hospital care, but it’s along the lines of the tone I’d want to go for.) Then, after the ad is over, we zoom out - turns out it was playing on the TV / computer screen somewhere in the office, and we’re dropped right into the avatar’s workday.

**What is the player’s role in the story? (the person things happen to, the person acting on things, the person building/shaping things, a bystander..etc..)**

The player’s role will be someone who makes decisions in reaction to what is happening around them in the game. The player is a member of a team which has a boss giving instructions, so the player does not have ultimate control over everything that happens. The only control that the player has is what decisions they personally make about their own behavior - these decisions can influence others (such as team members) to act in a similar way, or they might cause others to become upset. Ultimately, the team that the player is a part of has been instructed to always follow orders from the boss, so the player has a limited ability to convince others what to do. Types of decisions the player can make include whether or not they give the unfinished code to the boss, whether or not they talk to coworkers, and whether or not they go to the press with the situation that is unfolding in the office. These decisions are focussed around what actions the player individually takes, not what actions the whole team takes. Other team members may make actions contrary to the player (such as giving the boss their unfinished code whereas the player refuses to give up their unfinished code). The player will keep making decisions in order to reach an optimal outcome in the game. The player will have some level of control over how the story progresses as they can make many different decisions that all affect the story, but some important events will always occur every time.

The player’s impact on the story would be that the player is the decision maker. In a way, we are trying to make the player substitute him/herself into the avatar. I do agree with Professor Osterweil that the player should realize that there are different factors to consider and balance, such as coworker relationships and speaking your mind. However, the way that it should be presented should not be explicit, rather, it should be implicit through the hints presented in the text as the player progress in the story and after they make decisions to some prompts. I also agree with John that there should be a mixed of both linealarity and divergence.

I agree with everything that’s been said here! I also think that the player should have the ability to be a voice of reason, and convince his co-workers that something is awry if they ask them questions the right way; for example, if the player learns new information after talking to the boss, then the player ought to be able to divulge this information to co-workers in future conversations to potentially sway them towards caring more about the situation (and maybe even standing up in arms against the code being rushed, if the player manages to get a plurality of his co-workers involved)

**The Problem**

**What is the software that is being tested?**

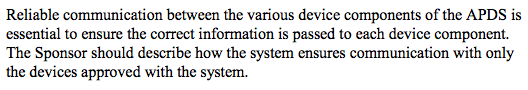
When I started my Unity prototype for the Testedness scenario, I settled on a component of some software for an artificial pancreas. (I chose it because this is a fairly recent development in the health sector, and because I was a little familiar with the tech, as my sister is a diabetic.)

As Professor Haas has mentioned, I think there definitely needs some extend of research that needs to be done about the challenging aspect of an artificial pancreas so that it can be realistic. Perhaps an idea could be the software that the sensory system uses to detect insulin level.

**What is the purpose of the software?**

This component of the pancreas’s software could be something relatively simplistic in nature (maybe calculating how much insulin would need to be deployed based off of a certain sugar level); if this component were developed insufficiently, though, it could spell huge danger for a user.

I definitely need to do more reading on it, but I think that figuring out *exactly* what piece of tech that the avatar is working on is an important piece of the puzzle. I found [this set of FDA regulations on artificial pancreas devices](https://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM259305.pdf), and this particular point stuck out to me when reading about the communication between the CGM (continuous glucose monitor, which checks the levels of glucose in your blood) and the insulin pump itself:



Maybe the avatar could be working on this communication component? There’s a lot that could go wrong in this transfer of information stage of the process (outside devices can illegitimately communicate with it, or maybe it doesn’t transfer data 100% correctly - not totally sure yet, but the possibilities for failure are definitely there!)

**Who are its most likely users?**

Type 1 diabetics, for sure! As of right now, at least, it wouldn’t be widespread in the diabetes community. Artificial pancreases are fairly new tech, and you’d either need to pay a fairly large amount of money to have access to it, *or* be involved in some sort of study.

The first artificial pancreas to be approved by the FDA (made by Medtronic!) [earned its approval in September 2016](http://wayback.archive-it.org/7993/20170111141252/http://www.fda.gov/MedicalDevices/ProductsandMedicalProcedures/DeviceApprovalsandClearances/Recently-ApprovedDevices/ucm522764.htm), so it’s definitely a pretty recent piece of tech.

**Who is mostly like to have an interest in how this software works (or if it works or not)?**

Both the software developers and the software users would be *hugely* invested in the effectiveness of the software. If a user wasn’t delivered the correct amount of insulin, they could end up in a diabetic coma. If that were to happen, the software developers would be in a *huge* amount of trouble - lawsuits upon lawsuits would rain down on them, and their artificial pancreas would *never* have a chance of being approved by the FDA.

I think the people in need of an artificial pancreas will have the most interest in this software and technology. However, since type 1 diabetes is common, I imagine that it would interest our society as a whole. I don’t necessarily agree that the software developers are always invested in the idea of the software they are developing, but they would care about the effectiveness of the software as part of their job. If a large corporation is developing this technology, then I would imagine it being different from an academic institution developing this technology.

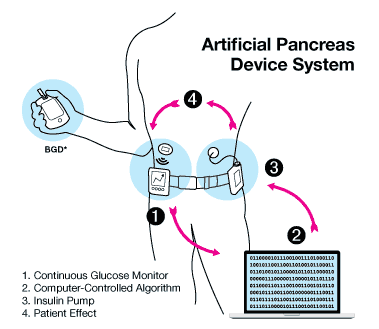
I’ve since read a little more into companies that make these medical devices - specifically Medtronics - and have found an instance of them [recalling the lead wire for their Sprint Fidelis pacemaker](https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm?id=65385)! (The lead wire is apparently the piece of the pacemaker that connects the pacemaker to the heart!) Apparently, the company (Medtronics) had minimal liability for the failures of these lead wires - there’s been some bills circulating (specifically the [Medical Device Safety Act](https://delauro.house.gov/media-center/press-releases/delauro-slaughter-introduce-medical-device-safety-measures)) that are trying to change the liability structure of cases like these, but as of now, they haven’t been introduced into law. It may seem a little off-topic to be discussing pacemakers, but I was trying to get some sort of a sense for how the liability structure works in cases like these; it seems that a failure of this software wouldn’t necessarily be a huge detriment to the company in the short-term (as it seems like it’d be difficult for a diabetic whose pump failed to seek legal action), but it might affect their long-term public image if word is to spread.

**Where (in what contexts) will the software be most likely deployed?**

This software would be deployed in pretty much *any* context - an artificial pancreas would be worn by a diabetic *wherever* they were, and glucose levels would be checked fairly regularly. (Every couple minutes, probably!) Insulin would be deployed semi-regularly throughout the day, too - this software would get a *lot* of constant use.

Ideally, the the software should be deployed after thoroughly tested with clinical trials.

Here’s a quick diagram of how the components of an artificial pancreas device works:



I got it from [an FDA article on these devices](https://www.fda.gov/medicaldevices/productsandmedicalprocedures/homehealthandconsumer/consumerproducts/artificialpancreas/ucm259548.htm), and it gives more explanation to how each component of the system interacts a little bit more in the article itself. The device is meant to be a personalized, mostly-automated thing, pretty similar to a pacemaker! The component of the system labelled #2 (the “computer-controlled algorithm”) happens on a “controller”, which could be, as Kyle’s said in his comment, a cell-phone-like device (or even a cell phone itself). So, with that in mind, this software could be deployed virtually anywhere that the software’s user will be walking around! (This context of deployment will be important - developers would need to make sure that each component of the system would be able to communicate reliably in *any* environment, even those with a lot of potential external interference - this would require a ***ton*** of testing.)

**What risks do you expect to be obvious to the player?**

It’s going to be fairly obvious that this is an important component of the software - if a pancreas doesn’t correctly calculate how much insulin a person needs, that person is going to be in some danger. However, a player unfamiliar with diabetes might not realize how fine-tuned this calculation might need to be, and how many variables influence how much insulin is needed.

If the player is referring the the actual person playing the game, the player should also be concerned about losing their job.

**Tensions**

**What other possible perspectives might exist about the software?**

Companies that produce products needed for manually checking insulin levels / intaking insulin might not want technology related to insulin to be released. When a machine can detect insulin levels and dispense insulin automatically instead of having the user do it manually, the user has to buy fewer needles or other medical equipment needed for checking / administering insulin. When people buy less medical equipment, companies that produce the medical equipment lose money (which is not something they want to do). This opinion is contrary to the normal belief that technology which makes people’s lives easier is a good thing.

^That is a good point. I assume that there is also hardware that is required for manufacturing for the artificial pancreas, such as a physical device to illustrate the detection result. Upon looking into this more, there seems to be some research projects about artificial pancreas, but the companies that are involved in it are the major biotech companies. I don’t think the companies would necessary lose money, because they will just be selling a difference service/product.

I found a really cool article that might be right up the alley of this project - i[t investigates some of the psychosocial and ethical implications](https://www.sciencedirect.com/science/article/pii/S1262363618300806) of artificial pancreas devices! One of the interesting things I found in there is that people might be nervous to use the system, as there’s a chance for malevolent actors to interfere with the data transmission devices in a way that could be used to harm the user. Another issue was dependency: if a user is incredibly used to automating the insulin delivery process through an APDS (artificial pancreas device system - I’m going to use this acronym going forward, as it’s common nomenclature in the literature), then their ability to manually give themselves insulin could totally depreciate; in this way, there might be a general air of distrust (or at least, a hesitancy to trust) these systems, meaning that a large-scale failure (due to lack of testing) could potentially impact the future development of these technologies. (Development of the pancreas is a huge technological milestone, but if there’s a massive failure and diabetics stop trusting the technology, then it’ll never be able to take off again in the same way)

Another interesting perspective (also brought forth in the article about the ethical implications) is that of access - apparently, people are concerned that these expensive medical devices will not be able to be shared with the right amount of people in a fair way. (This concern has even [lead to the creation of OpenAP](https://openaps.org/)S, an open-source, jerry-rigged version of an artificial pancreas system that’d be much cheaper to purchase than, say, the Medtronics version) I’m not necessarily sure how this might factor into the development side of things re: the avatar’s job, but it might be a factor to consider for a boss / upper-management stakeholder who’s trying to cut costs in some way in order to appeal to people who are concerned about the pricing of the device.

**What are some of the unexpected ways that this software might be used?**

The sensory technology and secretion of external hormones might be able to apply to other things, for example, the glucagon hormone for low blood sugar.

**What kinds of uses require intervention to prevent?**

If this technology could connect to the outside world (via bluetooth to the user’s phone for example), then some malicious person might want to hack into the system to disrupt the normal intake of insulin for the user / display an incorrect level of insulin in the body. Careful attention should be placed on making sure that the system cannot be infiltrated by hackers in order to protect the safety of the user. (I don’t know much about how this technology works but I am guessing that…) The technology may also restrict how much control the person has over how much insulin goes into their body. The machine is programmed to insert the correct amount of insulin in the body, but if the user thinks that they know more than the pump, they might try to override the normal settings for the pump and put more / less insulin in their body compared to the recommended amount.

I think some sort of self-detection debugging protocol would be required for this type of system warning the user about possibility of software malfunctioning and to contact for help rather than trusting the software 100% of the time.

**What are some of the consequences for unexpected uses?**

If an outsider is able to hack into the system, they would be able to manually alter the flow of insulin into the user’s body or alter the level of insulin displayed to the user. A change in insulin levels could negatively affect the user’s health because having too much / too little insulin in the body can cause adverse side effects. Also displaying an incorrect amount of insulin could lead the user to manually make decisions such as administering more insulin into their body without the use of the automatic pump. Again, this could cause the user to have more insulin in their body than needed which could cause adverse health effects. Also, if the user thinks that they know better than the pump, they may put more / less insulin in their body which could cause negative side effects if this new amount is incorrect.

Unexpected uses such as using the software when it is malfunctioning could trigger ketoacidosis, which can lead to death. I think for best software engineering practices, the system would always aimed to be secured.

Just a little elaboration on the potential for a malicious actor to influence the integrity of the system: apparently, since the FDA is primarily concerned about the effectiveness and the safety of the products they’re approving, they haven’t put a whole lot of stock into cybersecurity concerns about devices. [According to this article,](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4556085/) in the FDA regulations on artificial pancreases, the clauses about cybersecurity are only guidelines, and not regulations. So, there’s a bit of a lack of accountability if a failure like this were to happen.

**Why is testing important?**

Testing is important in (ideally) ensuring that problems related to the device will be found before they negatively affect users of the product. It may be a time consuming process for companies to test their products in every way imaginable to find possible flaws, but it could save lives when technology has a change of malfunctioning inside someone’s body. It would be less work for the company to release their unfinished product and have users submit reports of all the errors that they encounter with the product. However, this would mean that more customers encounter problems with the product as opposed to if the product had been tested internally before release. When customers encounter issues with technology, they will most likely be turned off from using that specific product if there are alternatives on the market that would work better. Some issues in technology can also cause adverse effects for the environment or the human body which the company that produced the technology (if they acted negligently) would have to pay reparations for.

A lot of softwares today uses agile for development practices, but for something like an artificial pancreas, it would require a waterfall development process, and even more thorough testing. The testing should be very extensive and adapt along with other research knowledges.

This is a product, too, that’d could be used in conceivably *any* environment; as Jon said, this product could potentially cause bodily harm, so even if the software failed in *incredibly* specific circumstances, it could potentially be deadly.

**Why is the timing of the testing important?**

Testing before the product is released is important in minimizing the number of users who experience issues with the product. Ideally, all of the issues with the product will be caught by the developers who will fix them before any user has to experience them. If little testing is done beforehand, then the users of the product will more likely be negatively affected by faults in the product.

I don’t quite understand this question. Is this question the same as the previous? Testing before release should always be done, but how much really depends on the software, but for an artificial pancreas, testing after release would extremely detrimental.

With certain things, you could take a MVP (minimal viable product) approach, release a barebones piece of software that *mostly* works, and then figure out further test cases later. This is *not* one of those cases, though.

**What kinds of tests should be run?**

For any piece of technology, tests should be run that measure the effectiveness of all different individual pieces of the technology, cover all possible ways that the technology could be used, and cover ways that the technology could possibly fail. This often requires creativity when identifying non-obvious uses or possible points of failure. Some possible tests for the automatic insulin pump would be: testing all aspects of the hardware (battery / motor / sensors / screen) to see if they all function properly in various scenarios such as different temperatures, altitudes, countries; if they function properly when inside a human (and many different types of humans); or if they unintentionally mess with other components. Also, the software should be tested to see if it can correctly tell how much insulin should be released inside a person, how quickly it can detect insulin levels and respond, if it displays the correct amount of insulin to the user, etc. All of the components of the technology should be tested together to ensure that the parts work as a whole compared to simply working individually. This means checking whether or not data transfers correctly from the insulin sensors to the decision making “brain” to the motors that dispense insulin to the screen that displays the amount of insulin in the body etc.

It appears to me that the academic institutions are able to recruit people for clinical trials for their studies, especially for the smartphone management systems that they are trying to build. I am unable to find how the large biomedical companies are currently doing their testing.

From my understanding, some of the biotech companies will [partner with organizations like JDRF](https://www.jdrf.org/impact/research/clinical-trials/) (Juvenile Diabetes Research Foundation) in order to recruit people for clinical trials. I think that Jon’s provided a pretty robust slew of things that a company might want to test an APDS on, but I’d like to bring up another one: penetration testing! In thinking about the communication between all of the individual components of the APDS, there’s some slight concern about any malicious actors who might want to intercept / change the flow of data. If the developers had this in mind, and went through some rigorous penetration tests, the chances of this happening are definitely much lower. Similarly to what Jon said about different environmental conditions, I think that’d be *hugely* necessary to test the performance of the devices in areas where there might be a lot of interference (say, a tech convention where there’s a *ton* of Bluetooth signals being sent out, or a large concert / sporting event where cellular data networks might be overly congested).

**What variations in testing might change the outcome in the perceived readiness/safety of the software?**

The testing should be planned very carefully for long term in order for it to be perceived ready. Any small bug that seems non-major should also be invested in and that could suddenly change the readiness level.

**Player Moves**

**What actions are available for the player to take?**

I think couple options that the player can make are:

Internal Options within the company:

1. Talk to your boss/manager that is managing the work you produced.
2. Talk to colleagues
3. Talk to someone higher than your boss or a general complaint office in HR department if there is one

External Options outside the company:

1. Go to the government
2. Go on social media (maybe anonymously)
3. Talk to friends that have also have similar jobs for advice.
4. Talk to close family members

Ultimate decisions:

1. Convince the boss and find another way around the problem from any above options.
   1. Maybe this can be accomplished through convincing other co-workers to take a stance against the poor engineering practices that the boss is promoting
2. Do what the boss requires and keep the job. (Most stable for the individual).
3. Ignore what the boss wants you to do, and do not resign (might get fired).
4. Resign from the job.

**When are those actions likely to have the greatest impact on the outcome?**

It depends on the action, ultimately, I think it would be very interesting for the player to have to do thinking (like solving a puzzle) in order to find a more desirable outcome. For example, some final options might not be available because the player did not collect enough information for it to be a valid option. Another uncertainty is rather we will allow the player to pick more than one option for some states.

I definitely agree with Kyle - I think that the main action the player will be able to employ is information gathering, so we should structure some of the information you can gather based off of how the player had proceeded. For instance: maybe talking to upper management before you talk to your boss will make your boss more likely to distrust you / shoot you down when you come to them at a later time.

**What influence does the player have over the situation (what decisions are within their power to make)?**

The player’s influence is having the technical knowledge of developing the software, it's unlikely that companies would want to replace a software engineer, especially if they are newly hired. I think the player that will be playing in the role of software engineering does have some degree of influence for the upper management to take the situation into consideration. The player’s ultimate influence is that he can choose to quit the job.

The player will not be able to make any direct decisions over the way that the product is released, but they will be able to potentially change the minds of their boss / co-workers.

**What are the stages in this story where those decisions happen?**

Information investigating types of decision will most likely occur in the middle of the FSM while the ultimate decisions occur near the end.

**What knowledge about the situation will the player need to take action to reveal? (In other words, what will the player not know until they take some kind of action in the game?)**

The player will not know how their coworkers, boss, or anyone else feels about any given situation until they talk to them about it. This means that talking to people will give the player more information that they can use to make the best decision.

The player and coworkers might all have a baseline knowledge of how Diabetes and insulin works based on workplace training or outside experience. Training could allow them to realize how important it is to ensure correct levels on insulin in the body. They also might not receive any training about what Diabetes is because they really don’t need any medical training to be able to make this automatic pump. Each programmer will have to make a small part of the machine such as something that senses insulin levels or starts a motor when a certain insulin level is reached. They don’t need to know anything about Diabetes to make those mechanics work properly. Therefore, none, some, or all of the coworkers will understand how Diabetes works. The level of knowledge about diabetes might affect how much effort and care the workers put into what they produce. For example, someone might not understand the disease very well which would make them more likely to not care as much about the final product. They would then be more likely to release software that has not been tested thoroughly because they don’t understand the severity of the disease. The player might have a general understanding of how his colleagues feel about the disease / product, but the player will need to talk to individual people to understand fully how they feel about the software potentially being released without being tested thoroughly. I would assume that all workers would have a general understanding of how there could be legal consequences if people get injured by the technology they are creating, but I am not sure to what extent the workers would know.

Depending on the parameters set, I believe that the “NPCs” would respond differently and therefore the player would not have those knowledge before interacting with them.

The player also might not know how expensive a APDS are, or what motives the upper management might have for trying to push this technology out faster. Maybe the player might not even think of the possibility of someone’s glucose data being intercepted from the device, but conversations with others reveal this to be a potential.

**What kinds of interactivity do we want to see happen?**

We would like for the player to be able to talk to their coworkers, boss, or anyone else in the company and have these conversations affect the outcome of the game. The player might also talk to people outside of the company such as legal experts, doctors, people with diabetes who are potential users, their friends / family, or anyone else that could give them useful information. Talking to a potential user would give information about how useful the product could be and show the player that their technology will be affecting real people. Talking to a legal expert could inform the player that they might be liable if the technology injures or kills a person. Talking to coworkers might inform the player if any other people agree with what they want to do (which could affect their decision making). For each decision that the player makes, the player will be prompted with several choices and they must click on the choice that they wish to proceed with. Knowing more information from talking to people will affect the the choices that the player makes as they become more aware of how their coworkers, boss, or anyone else feels. After particularly important choices, there will be a text box in which the person playing the game can record the reasons why they made a certain choice.

I’m not referring to if the player will be able to move around or what type of things they will be able to click on. I’m more referring to what types of general decisions the player can make in the game to further their progress.

^I’m a bit confused if John is referring to an actual movement of the avatar in the game or just a change of scenes.

I think Jon is avoiding commenting on the moment-to-moment *actual* movement of an avatar within the game, as we don’t know what degree we might need to pursue that. Personally, I do think that the ability to walk around the office might be a good thing - if this were the case, we might even to be able to hide pieces of information in the environment for the player to find. (Maybe, for instance, a couple letters from people who absolutely *can’t wait* to participate in an APDS study, or some flyers about the impact of diabetes.) Even if these pieces of information weren’t game-changers when it comes to some of the conversations a player could have w/ their co-workers, they’d do a good job at further providing context for the situation (and an idea about how important of a product this is)

**Outcomes**

**What is an ideal outcome from an ethical perspective?**

An ideal outcome from an ethical perspective is that the player utilized critical thinking skills when making decisions in the game and thought about the potential consequences of each decision / how each decision could have gone wrong. This could be shown in the form of how long the player’s responses are (when they have to type responses). To me, it seems like longer responses would mostly corrollate to thinking more about the decisions, but that could be a wrong assumption. There could also be a tracker to see how much of the environment the player explored. If the player explores more / talks to more people, then it might show that they have a greater interest in the story and would have thought about their decisions more.

Idea situation from an ethical perspective is to not release the problematic software and even prevent it having the opportunity for someone else to release it.

Ideal situation for the game developers(us) would be that the player discovered more than 50% of the entire game. (One playthrough might roughly unlock 15-20%).

Like Kyle said, I think the most ethical thing to do in this situation would be to *try* to delay the release of the product in order to perform further testing. Obviously, as the avatar is not a senior decision maker at this company, the most that they can do is try - they’re not making the final calls, so it’d be unrealistic to assume that they’d be able to prevent an unreasonable boss from still pushing this software through.

**What is an ideal outcome from a player perspective?**

An ideal outcome for the player is that all of their responses, decisions, and other data collected throughout the game have been recorded. This data should be viewable to only the teacher (besides the student) without the student having to do much work. In my opinion, the less work the player has to do, the more they will probably enjoy the experience. They player might also not want to have to complete the simulation multiple times, so only completing it once might be ideal.

It really depends on how invested the player is in this game. From the player/student’s perspective, the primary ideal outcome for them is to just get the credit for the assignment.

I think a player that isn’t invested in the game would ideally just want to finish it, so they might just try to get through it as quickly as possible. If someone *was* invested, I think an ideal outcome for them would be some sort of “win” state (i.e. the boss relinquishes and lets you develop further tests for the code).

**What outcomes are possible?**

The player can explore any number of unique paths (from making different decisions) as they complete the scenario.

**What does a negative outcome look like?**

A negative outcome would be if the player does not put much thought into their decisions and completes the game simply to get it over with instead of trying to learn something. The player might not put a lot of thought into their answers for several reasons such as they found the game uninteresting (from the game being too easy, they couldn’t connect emotionally with the character, etc.) or did not think the game would be worth their time to pay attention to. The player’s level of engagement with the game can be measured by the amount of time it takes for the player to complete the game, how long their text responses are, how many different paths they went down in the game, etc.

**How will the player know when they are nearing an outcome?**

There will be a progress bar present throughout the game that shows the user how many different paths have been visited in the session since the player started playing the game. This means that the user can replay the game any number of times in order to increase the overall score of how many different paths they have come across. At any time, the user will be able to see how many different paths they have visited, so they will always know what outcome (how close they are to visiting all paths) they are at. A player will know that they are nearing a new outcome when they have to make a decision in the game that could lead them down a path that hasn’t been visited yet.

I don’t think it is a good idea to show any sign/progress for the player to know when they are nearing an outcome (one ending) because it produces a sense of how much time they need to invest in (which might cause quick clicking through without thinking) rather than letting them get attract to the game itself.

Since this whole scenario is about rushing code in order to hit a deadline, I think there ought to be a certain amount of “days” that the player has to explore the situation. Certain situations might trigger a final outcome faster than others (for example, if the player were to decide to go to the New York Times after two days of investigating), but generally, the player would know that their deadline was in a couple days. My idea of the flow of the simulation is that the player is able to take a number of different actions on any given day, with certain events always happening a certain amount of days into the simulation / certain options only opening up at certain times. Then, once the player decides that they’ve done all that they want to do in a given day, they can move onto the next day to progress the story; in this way, they’ll always be counting down towards an eventual deadline, where *something* has to happen by!