https://courseworks2.columbia.edu/courses/53313/assignments/145734

General Comments:

1. Overall some excellent PRDs, some good and some that need some additional love
2. Remember - PRD is
    - ***Why & what*** (why/what problem are we solving and what would we build in terms of feature sets/requirements. No mention of how/tech)
    - **not**
    - ***how*** (how will we build this / how will we *implement the requirement* in technologies)..

    I appreciate some of you have spent many years studying ***how*** and this has built a reflex to think in terms of ***how*** to build. I have emphasized in lectures it takes some people years in industry to break out of this reflex.

    Think of it this way:

    - If I say ‘how/implementation and not ‘what/requirements then I have given a solution for an undefined problem!
    - If I am sent a design at work I can not evaluate a design. A design is a thing. It is a ‘how’. My questions are always “does this design meet requirements?”
    - There are many ways of solving a single ‘what/requirement’. I could use many many different technologies or designs. By specifying ‘how’ so early I may be cutting my design space / options. If I focus on requirements I retain as many design options as possible.
3. You should be able to give this to someone not taking the class (or even the user) and they should be able to skim through it and say “this make sense”.

Section 1:

- **Why**

Comment 1:

Many of these were really great.

However, a few of the PRDs just launched straight into “we will create a system which does X”. Their section 1 described *what* would be built but not *why*? Without looking at paper or a keyboard turn to your teammate and ask “why?”. Whatever they say should probably go in the document. If they give you a technical ask “why?” again until you get to the bottom of it or drive them nuts. Whichever comes first!

Comment 2:

In a second set of PRDs they then went *back* in the 2nd/3rd paragraph and added the why. The format was therefore,

1. We are going to build an product with these features
2. The reason we need to build this thing is to solve a problem in the world

Although we got there. For a fresh reader reversing it is easier to follow

1. The problem in the world is X and a tiny bit of context
2. As a result we propose a product which does Y
3. We think this will provide value

Please read and understand the curse of knowledge. Basically, when you have been working on a problem for days you are so deep into it and assume the other person knows all these issues that your explanations assume too much. Feel free to test your PRD on a family / friends / student outside the class. They can’t write it but even without having taken the course it should make enough sense to them.

Comment 3

Even if you are producing a game please state *why* you are creating the type of game. Is it because people love X and Y but no two games mix those two gameplay features? Is it because programming skills are key for future generations but much teaching of programming to kids is dry and or requires installation of complex tools? You propose a game where …… etc.

Comment 4

1-2 (and it may be draft or reorganizing) have very specific functional or non-functional requirements in the intro. Look through your intro and see if you are specifying functional or non-functional requirements. You don't need them here and there is a section later (i.e. no need to specify performance requirements or seconds in the intro)

Section 2:

- **Who is the product for?**

Comment 1

The teams who read the template (provided on the assignment) and looked at the exemplar from lecture #2 did very well. The teams that seemed to be less familiar with both had more trouble ….. :S

Comment 2

A few of the teams who used the template have merged *goals* and *user stories.* The goals are meant to be slightly broader.

"[goal] describes the problem the persona wants to see solved or the benefit the character wants to achieve"

This is not the same as a user story which is a lower level feature/piece of functionality the system wants to solve. For our bagel shop example

- The goal for a bagel buyer is something like "save time on the morning bagel run and get piping hot bagel".
- Their *user stories* would be "As a bagel buyer I want to be able to look at the menu so I can work out what to order". And there would be *many* user stories.

Please look at your persona goals. If you are specifying stories or even technical statements about databases or flask you have slipped too low level.

Aside: Sometimes a user may say I want a "database" of information. What I find is that they are providing a requirement in technical terms and it is actually dangerous. What my users actually want is "a single consolidated way to access information X,Y and Z and do A, B and C". "But Ewan - that sounds very much like a database!". However, I may choose to put video files in S3 and financial records in relational/Postgres. Therefore, if I separate the *requirement* (ability to access information) and *implementation* (how we build it) I find I have more options. A technologist who accepts

a requirement of “a single database” may now have less flexibility and a lot of problems fitting different types of data into a single database!

Comment 3

From a formatting perspective some teams put the user stories with the personas. I don’t mind as long as we dont confuse goals and stories (see comment 2 above). However, remember they are related exercises

- Persona: to create a profile of the person using the software to understand them better
- User Stories: identify, capture and document the functionality they require

Section 3:

- **What do the personas need to be able to *do?***

Comment 1

There are a few different templates/formats for user stories. However, the difference is minor for our purposes. The format we suggested is ….

`As <persona>, I can <what?> so that <why?>`

Please look at each of your user stories and ask if it fits the format. The most frequent issue is teams missed the whole <why> clause!!. For example they wrote (strike through for emphasis this is not what we want)

~~`As <persona>, I can <what?>`~~

Remember when we role played scrum in class and the product owner read a card to the software engineer who replied “what do you mean by ‘menu’”. The why is important

1. So the team knows the *value* of this story (and therefore helps prioritizing)
2. It provides context to the development team

Comment 2

We care that the *requirements* are captured. As noted in lectures there are *multiple* tools and techniques for documenting requirements. Most of the teams stuck with user stories. If you did pick another technique then please make sure you have captured the features in enough detail. One team opted for a what I (think) is a use-case diagram.

However, although the diagram mapped personas to use-cases I was not clear what the use-cases actually were.

Aside: there are so many different *views*/types of diagrams in software engineering. We will meet them very shortly! If you include a diagram *always* include a title - what *type* of diagram is this? Also say which viewpoint level the diagram covers. (We will hit this in lectures after midterm)

Comment 3

Remember - user stories are not ‘tasks’ in the engineering sense yet! We will convert the ‘what’ of a requirement into the ‘how of engineering’ very shortly! Again, don’t let ‘how’ thinking infect ‘what’

Aside: Sometimes if I have a problem with a team who have not captured their requirements and are stuck in design (they are actually stuck in design because they don’t know what they are designing for). I will be very blunt and say “Ok - can people humor me - I want to ban us all for 30 mins from talking tech - let’s talk product which is ‘why are we doing this, who is it for, what do they need to be able to do, what are the key non functionals’”.

Section 4:

- **What are the key non-functional requirements?**

Our challenge is that in this course we can not spend N weeks on the engineering discipline behind each non-functional. For example, in class I noted that for availability there is a set of terms and theory to capture (RTO, MTTF, MTTR, etc). We have not covered this so I am happy for you to go through the list and acknowledge which non-functionals apply to your project and discuss the requirements in non-engineering terms. Most of the teams did a good job of this but there were a few wobbles.

Comment 1

Here is a ‘what’ non-functional

Latency	For the game to be smooth running we want to hit 35 FPS
Capacity	Must support 1000 concurrent users

Here is an example of ‘how’ thinking:

~~Latency~~	~~We will use GeForce 4400 card~~
~~Capacity~~	~~We will shard the data if required and cluster databases~~

Hopefully, this makes ‘what’ vs ‘how’ very clear! 50 FPS is a requirement. There are many many many ways we could hit that FPS. We could write very efficient software. We could reduce the graphics complexity. We could use hardware acceleration. Depending on what we are building it may not even be an issue / require specialist attention but we have established it as a requirement.

However, if we state requirements in *‘how’ language* our question is “Is that necessary?” and “wait - what is the *requirement* we are solving for?”.

Takeaway - think *requirements*(what) not *implementation(how!).* It will save you time!

Comment 2

A small number of teams leaked functional requirements into the non-functional section.

Section 4:

- **Prototype how the product could look/work**

Generally good. If you are using storyboard you may want to show the navigation between screens. With 5-6 static wireframes you may not all agree on how the users navigates the app to achieve a customer journey.

Section 5:

- **Produce a roadmap**

Generally good. Again, teams that borrowed the template seemed to have an easier time.

Comment 1

I am not asking for a detailed waterfall plan! If you have laid out a development schedule (requirements, design, construction, implementation) then you have created a waterfall plan. Now we have covered Agile you will recognize the PRD is an *envisioning* technique.