Hello all, I am Brandon Hobbs and live in Houston, TX. Houston is the 9th place I've lived since graduating Texas A&M almost 20 years ago with a degree in Mechanical Engineering and Business. Since then I have been in the oilfield in many different parts of the company.  
  
Currently, I am the head of Stimulation Design Software in my oilfield software group. Even though I am a self taught "computer guy" I thought it prudent to formalize my education - so I decided to enroll in a BS in Comp Sci with a Data Engineering focus. I have designed and built a few applications for the company - kinda scary.  
  
I am comfortable with C, C++, C#, Python, VB, VBA, SQL, and a few others. What I am not comfortable with are high level architecture decisions which I blame on my lack of "basics".

I am most worried about the time commitment part of going back to school and the balance of work/school/life. My plan to book certain parts of the day for school and alternate courses per day. Plus, I'll use the resources I have in my office to stay focused (typing this on my lunch break).  
  
So far my favorite course was either MATH350 or CS-300. In both I enjoyed the challenge of thinking a bit differently.

* What are the major phases of the systems development lifecycle (SDLC) Describe the key features of each phase in your own words.  
    
  The key parts of SDLC are planning, analysis, design, and implementation/maintenance.  
    
  Planning is exactly that. Gather enough information to know the what, how, and why for the system.  
    
  Analysis is where you actually try and understand the what, how, and why.  
    
  Design is where you turn your understanding into action. In the design phase this is primarily still "on paper". Mock-ups and prototypes may be created to refine ideas.  
    
  Implementation is where ideas are turned into actions and something is built.  
    
  Maintenance is a support activity that allow you to refine and smooth any rough edges of a previous design.
* Requirements can be collected in various ways: reading documents, an in-person conversation, joint application design (JAD), etc. Choose one of these methods and describe a strategy you would use to help you collect requirements with this method.  
    
  In an in-person interview you can try to ascertain what that individual believes to be good and bad about a current system. One important thing to achieve is to keep them talking but challenging them at the same time. A common technique is the 5 whys, i.e., asking why 5 times. This way you can try and distill the kernel of truth not just the interviewees perception of why.
* How has technology changed how people collect requirements?  
    
  A few ways I use technology is UserVoice and usage metrics. Both systems provide me with a direct connection to my actual users. UserVoice allows users to provide their ideas for improvements without being prompted by the engineering team - simply prompting someone can change an answer sometimes. Usage metrics shows me exactly how a user interacts with my systems - without context however. Without this context adding your own version of the story is easy so interviews are still needed sometimes.
* What ethical issues (data collection, bias, and so on) do you think need to be considered when collecting requirements?  
    
  As before, usage metrics can be gathered by the system during usage. It's imperative try and keep this as anonymous as possible and still be useful. What you wouldn't want to do is to start to violate a user's idea of "fair usage". For example, tying a user's usage with their personal data to try and identify who they are and then interrogate them on why they did something would be frowned upon.

**RESPONSES**

**1**

An interesting phenomena of interviews is that the answer can change depending upon the observation technique. That is to say, if you give the user a written questionnaire with no time limit you may receive a different response than watching them physically use the software.  
  
I will say that having someone try and replicate their own process why asking them "why did you do that" can be very enlightening - and Zoom and other screen sharing tools facilitate this well.

**2**

Sometimes you can actually get better data when it's anonymous. Watching someone can change their answer as they may want to prove something to you or try and "do the right thing". Another good aspect of anonymous statistics is that you may not fall prey to the Personally Identifiable Information (PII) rules. These rules, and laws, can be very onerous and prevent you from actually using data.  
  
There is a limit to how anonymous data can stay and still be useful, however, As you say it becomes very easy to add your own story/bias to numbers.