Summarized and themed questions and answers:

**Theme 1: Technical Debt Identification**

4. Stages in Software Development and Key Challenges:

* The participant discussed their role as a developer and how they start with the tools they have but learn new ones when needed.
* They highlighted challenges related to using different frameworks and syntax when integrating additional tools.

5. Identification of Technical Debt: The participant mentioned that they become aware of technical debt during debugging, which happens at the implementation level.

6. Indicators of Technical Debt: The participant mentioned that indicators or red flags for technical debt appear when they push information from the front end to the back end, and they find discrepancies or issues during integration.

**Theme 2: Technical Debt Measurement**

7. Measurement of Technical Gaps: The participant primarily

relies on version control systems like Git for measuring technical gaps.

They mentioned not using specialized tools like SonarQube or Step Size for code analysis.

8. Current Tools for Measurement: The participant confirmed using Git for version control but indicated no usage of specific tools for technical debt measurement.

9. Prioritization of Technical Gaps: The participant discussed prioritizing by focusing on the minor issues first, as they can often be resolved with limited resources. They suggested leaving major issues for later.

**Theme 3: Technical Debt Impact Evaluation**

10. Impact of Unresolved Gaps:

* The participant described how unresolved gaps, such as bugs or lack of documentation, can negatively affect the development process and productivity.
* They emphasized the importance of understanding code to avoid technical debt.

11. Specific Examples of Impact: The participant shared an example of a technical gap affecting the project, where they couldn't understand the behavior of a reporting feature.

This led to additional work and time spent resolving the issue.

**Theme 4: Early Debt Repayment**

12. Practices to Encourage Early Repayment:

The participant mentioned that there are currently no specific practices in place to encourage early repayment of technical debt in their college or curriculum.

13. Incentives for Managing Technical Debt:

The participant discussed the incentive of seeing peers complete tasks early, motivating them to do the same.

They expressed a desire for students to submit their work early.

Insight

14. Additional Insights:

The participant suggested that documentation and readme files should provide clear and concise information to help developers understand frameworks and tools more easily.

They mentioned that frameworks should be designed to simplify implementation.

**Full Trancription**

Certainly, here is a transcription of the participant's responses you provided:

Participant: I will start by introducing myself. My name is Mugoya Dihfahsih and I'm doing a master's degree in software engineering and data communication from Makerere University. I'm carrying out research in finding solutions or ways to minimize technical debt in student prototypes, like final projects and early software development teams. I would like you to take me through your project you're doing and then your role on it, and then maybe what type of the project is it. But before I go into that, maybe I can bring up to date what technical debt is. Technical debt, these are the repercussions of taking shortcuts when developing a software prototype, and these shortcuts, with time you have to repay them. So basically that's what technical debt is. Maybe you tell me your name, the course you're doing, and maybe the project you're working on.

Participant: I am Kyeyune Habib. I'm doing research in front of the idea. The project basically, it is like for guidance of the students that have reached a stage of going outside to internship places. This is a simple system that will help them to find places that will be beneficial for them, where they will put most of their knowledge that they have gained from the classes, and put them to use. Still the system will help them compile whatever they are required to handle after attending the internship. And from that point, a compiled document to act as a source for information while documenting their reports.

Participant: So what role do you normally play in the project you're working on?

Participant: I try to say personal projects.

Participant: Personal projects?

Participant: Yes.

Participant: So you work as a developer?

Participant: Yes, I work as a developer.

Participant: Okay, so that's good. So can you describe for me maybe the prototype you are currently working on, including is it a web app, is it a mobile app, is it a desktop application?

Participant: And the intention, like you mentioned, the functionality you have mentioned, but maybe is it a mobile app or a web application?

Participant: Yeah, it is, basically it's a web application.

Participant: Okay, so what are the main objectives and the requirements of that prototype?

Participant: The requirements in developing, there are a lot. But for developing, the requirement is all about adding it onto another system that's already running, so that it's added as another function to be explored by students. So the requirement for students to use it, it's all about the registration number and the details that the campus that will be using it uses for the identification.

Participant: Okay, so this research is basically going to be divided into four themes. We are going to have a theme of technical debt identification, then the measurement, the impact, and then the repayment of technical debt. So the first part is, in your experience, what are some of the stages involved in developing a software prototype and what are the key challenges that you've encountered that are related to the tools you are using, the standards, the framework, the programming languages, and the conventions.

Participant: What are some of those challenges and what are those stages that involve yourself when you're developing a prototype?

Participant: As a developer, personally, when developing, I head on to develop with what I have currently. But as time goes on and the system requires additional tools, then if it means learning about the tools, I put in the time, I learn about the tool, and I come back and implement it into the system. If all fails, then I seek guidance from those that have already used the tool and are well-versed in it. Because, you know, it does not stop. Yeah, true. You might find someone who is already a member of a cultural organization and you have taken some good on it without knowing how to employ it into your system.

Participant: Yeah, so what are some of the key challenges that you normally face when you are doing that?

Participant: Culture, yes. As in, what exactly is it that you want from these guys?

Participant: Yes, it's an interview where, basically, it's a software architecture. Like, the gaps that we can identify in their projects and then how do we minimize those gaps in the final product output.

Participant: So, can it be like a collective effort to talk to yourself, students?

Participant: It's okay, but this one, basically, I'm looking at their individual input in a project. But also, it can be like a focus group. We can make the most of it.

Participant: Okay, so, what are some of the challenges you mentioned there?

Participant: I was speaking of challenges.

Participant: Okay, most of the programs, they use the same syntax, the basic syntax. Although, when you try to use something like a framework, then the framework is quite different. Sometimes the framework carries new stuff for you that you have to use.

Participant: Okay.

Participant: So, basically, if you use that framework without understanding, then you are like a technical debt because you have to redo what you never understood, right?

Participant: Yes.

Participant: So, how do you normally identify or become aware of the technical gaps in your project?

Participant: Actually, that is at the time of debugging.

Participant: At the time of debugging?

Participant: That is at the implementation level?

Participant: Yes.

Participant: Okay.

Participant: So, what are some of those indicators that suggest that your project or prototype has a technical debt?

Participant: Like indicators, like things that show in your project that you have a technical debt or you need to refactor your code.

Participant: When pushing information from the front end to the back end.

Participant: Yes.

Participant: You might find out that you are pushing information, but what is supposed to be refactored is not what is there.

Participant: So, basically, that is at the integration level, when you are integrating the code?

Participant: Yes.

Participant: Okay.

Participant: So, let's look at number two, that is for technical debt measurement.

Participant: So, how would you like to measure these technical gaps in your processes?

Participant: How would you like to measure them? Because, first of all, you have identified them maybe at the integration level, at the implementation level. So, how would you like to measure them and quantify such that you can easily maybe prioritize what to deal with first? Because in the measurement, we normally have analyzing using maybe version control, because I saw you with GitHub, maybe contacting the stakeholders, maybe calling out code reviews, code analysis, and code issues, like identifying the issues in the code.

Participant: We have tools that we use, maybe you can use step size, Sonar, SonarQubes. Have you ever heard of such tools?

Participant: No.

Participant: So, that means you normally use only Git to analyze your code, right?

Participant: Yes.

Participant: Okay.

Participant: So, for you, basically, measuring the technical gaps in your product, you normally use version control.

Participant: Okay.

Participant: So, which are the current tools that you normally use, or you said you use only Git?

Participant: Yes.

Participant: So, you've never used the SonarQubes or Step Size and code requirement?

Participant: No, not yet.

Participant: You've never heard of them?

Participant: No.

Participant

: So, those are tools that we normally use to identify technical gaps in our processes or prototypes. Basically, you can put them inside your code, the extensions that you can use to identify the gaps.

Participant: It gives you a familiarity. This class is too long. Please break it down. It may cause you a technical debt, right?

Participant: So, you said you are a software engineer?

Participant: Yes, software engineer.

Participant: SonarQubes?

Participant: Yes, it's called SonarQubes.

Participant: Then, there's code requirement. Then, we also have Step Size. Step, is it? Step, like step size. Step size? Yes.

Participant: So, these are extensions?

Participant: Yes.

Participant: Can I implement them in VS Code?

Participant: Yes, you can just install them in VS Code. They enable you to identify the leakages in your code that will lead to technical debt.

Participant: Alright.

Participant: Okay.

Participant: So, how do you normally prioritize technical gaps to address first?

Participant: How do I prioritize?

Participant: Yeah.

Participant: How to strengthen my software? The system. If the project has two sides, it has front and back. First, concentrate on the ICU design. How does it… How will it be… Automatically, different devices.

Participant: Okay.

Participant: Like, maybe what I meant is, how do you normally prioritize? Like, maybe say, you have a system. You have maybe errors in the back end, errors in the front end, errors in maybe database, something like that. Now, you want to address those errors. Which one do you normally consider? Is it the one that you are so critical to the system? How do you measure? How do you normally address the errors first in your system? Maybe even the minor ones.

Participant: Yes, the major ones, right?

Participant: The minor ones.

Participant: The minor ones.

Participant: So, you leave the major ones.

Participant: Or even find them a very big time to concentrate on.

Participant: Oh.

Participant: Most times, the minor ones, you might be able to solve them with limited resources and they work out.

Participant: Okay.

Participant: That's great.

Participant: As long as it meets… A lot of time. Yeah.

Participant: Okay.

Participant: Okay.

Participant: So, let's go to part three where we have technical data impact evaluation.

Participant: So, in your software development, how does these unresolved gaps affect the quality of your software? Maybe in terms of reliability, performance, and maintainability. How do those gaps… Maybe the gaps I'm talking about, maybe they could be bugs. Maybe documentation. Maybe you're using shortcuts. Those are called suboptimal solutions. You've heard of chat GPT. So, maybe you get solutions from there and you implement them in your code and they work. Your project kind of works, but with time, you can't really understand what you were doing, right? Of course. So, how do those unresolved software gaps in your prototype affect your development process? Most times, I don't develop this project like constantly, I say day to day, every day. Sometimes, I might develop this week. Next week comes, you have a test. You have to leave it for that week. Then, by the time you come back the other week, you might find that you want to add something to the top, be acting with something at the top, but what is at the top, you didn't put anything like a note. Yeah, yeah, yeah. Documentation. So, you have to go back and pass through the whole code as you implement it to see the reactions, until you find what you're going for. Also, you always have those, you have to go back to understand what you are implementing before you add in a new feature.

Participant: Okay, so, basically, that is performance-wise, you are not performing, you are not productive, right?

Participant: Okay, so, can you provide maybe a specific example of how technical gap affected your project outcome?

Participant: Okay. For example, we had just begun to touch the backend.

Participant: Yes.

Participant: The system, and how to bring them together. There was reporting information, but now, you don't know what it's bringing out. You're putting a name, and it brings out your value from the same source. Now, you ask yourself, why is it coming like this? Now, you have to look for someone who knows it, but you might find that your colleagues are also, they use some other steps to pass through it, and you use something different. Now, you find yourself, you're left alone, and you have to head out. You ask GPT, GPT gives you the usual stuff. Now, you have to go deeper and deeper until you find out the solution. Then, it takes a little time, finding out the solution. It is just a minor thing. You exchange your value, you put it to something else, and that will give you the good output.

Participant: That's cool. That's great. Actually, that one brings in the impact of technical data. It's within us, though we don't know that we really have it. But when you are aware of it, then it gives you time, and you're like, now what I'm calling is now the software, the technical data, maybe that I would have solved a long time ago. Instead of just using this code and get through with it, I should have maybe first understood what it's doing. When you understand something, it's always with you forever. You never forget it, at least you have an idea. Rather than just using it and you get away with it, it will always come back haunting you. Maybe you've used the code to pass the exam. You're employed somewhere, they give you the same code, but you can't recall what it was doing because basically all you did was just pass exams, right? That's what the technical data is all about.

Participant: So maybe we look at debt repayment, you know, like in the loan. If you take a financial loan, by all means you have to repay. And you always repay with interest, right? So also technical data is in the same way that whenever you take the shortcuts to develop your project, by all means you have to repay. Sometimes it could be costly because maybe you've implemented something, and then the supervisor is like, you never collected enough resources or maybe requirements. Then they're telling you, maybe you're at the implementation level, and they tell you, you go back and collect more requirements. Are you saying that you're being affected? Yes. So are there any practices in place, maybe at the college, or in the curriculum currently you're using that encourage early repayment of technical debt? Currently there's nothing like...

Participant: Okay, then that's fine. Then what are some of the incentives or mechanisms that exist that you know of that motivate you to manage or reduce these technical gaps?

Participant: The fact is that I see my colleagues completing their hours on a daily basis, which is a short period of time. And I am not going to... At an earlier time, before time, whichever. I would like all of these students to do a hand in.

Participant: Okay.

Participant: So, I think, what can be some of the insights or things that you would like in this framework, such as a framework to help students or early software development? What would you like such a framework to include?

Participant: Pardon?

Participant: Some of the readme.

Participant: Okay.

Participant: Some of the readme.

Participant: But something good with readme is they don't give you... They don't direct you to read what you want to read. You have to read the whole thing to understand how you're going to use it. If it is like... Some days, a framework is called... I mean... Something like that. The phrase is there. What you want, it's already known that this is for this, this is for this, this is for this, this is for this. You just hit it, implement it, and you're done. We hope those errors... I got you two things. I think some of them are so hard to find out. It will give you everything, so you have to join whatever it has to give you. To come up with what you want. And sometimes it may not be like 100% useful. So, you know, like, they want you to have two forms. If you want a function that does something specific to what you want, you might find out that you will have to write it down.

Participant: Thank you so much.

Participant: And it's also constant.

Participant: Oh, thank you so much, by the way, for the information you provided. It's so important, actually, to hope this research to come up with the guidance that enables students to have projects that have minimal technical debt. Yeah, I thank you so much for your time. And I just really need you to maybe to sign on this.