Participant 4: **Roland Kizza**

T**heme 1: Technical Debt Identification**

4. In response to the question about stages in software prototype development and challenges, the participant

* mentioned that the stages include requirements gathering, design, implementation, testing, and deployment.
* Challenges arise concerning tools, standards, frameworks, programming languages, and conventions, particularly when integrating third-party libraries.

5. When asked how they identify technical debt in their project,

* The participant explained that they primarily rely on code reviews and team discussions.
* Technical debt becomes evident when code complexity increases or documentation is lacking.

6. Regarding indicators of technical debt, the participant pointed out missed project deadlines, increased end-user bug reports, and the need to take shortcuts without addressing underlying technical issues as key signs of accumulating technical debt.

**Theme 2: Technical Debt Measurement**

7. In response to measuring technical gaps, the participant

* Mentioned assessing code quality using metrics like code complexity, duplication, and test coverage.

8. When discussing tools for measuring technical debt, the participant

* reported using SonarQube and CodeClimate to gain insights into code quality and identify areas requiring attention.

9. Regarding prioritization of technical gaps, the participant explained that they consider factors such as end-user impact, issue complexity, and whether the issue hinders other development tasks.

**Theme 3: Technical Debt Impact Evaluation**

10. When asked about the impact of unresolved software gaps, the participant stated that these gaps significantly affect software quality attributes. For instance, performance bottlenecks can result in slow and unreliable software, leading to a poor user experience.

11. The participant provided an example of how technical debt affected their project. They mentioned that neglecting to refactor legacy code caused difficulties in adding new features, resulting in missed deadlines and user frustration.

**Theme 4: Early Debt Repayment**

12. In response to encouraging early debt repayment, the participant reported implementing regular code review practices as an effective strategy to catch issues early in the development process.

13. When discussing incentives for debt reduction, the participant mentioned a recognition and reward program for team members who contribute to reducing technical debt. This program fosters a culture of proactive debt management.

**Insight:**

14. The participant emphasized the importance of prioritizing the identification and addressing of technical debt throughout the software development lifecycle. This proactive approach enhances software quality and reduces long-term maintenance burdens. Monitoring code quality metrics and fostering a culture of debt repayment are essential for smoother development processes and improved end-user satisfaction.

These responses provide concise points in reported speech format for each themed question. If you have any further adjustments or specific requirements, please let me know.

**FULL TRANSCRIPT**

I'll take you through. Yeah, I'll do that. Okay then. So, my name is Mugoya Dihfahsih. I'm doing a Master in Software Engineering at Makerere University, here. And the basis of my research is to identify the metrics for measuring technical debt in student software prototypes.

The most especially by finance students. And I'm here to speak about technical debt. Technical debt can be understood as the cost process for taking shortcuts or making compromises during the software development processes. This is comparable to a financial debt in the sense that it will accumulate interest over time, making it hard to maintain and upgrade the software in the future.

Also, technical debt represents extra work and challenge work that arise when code is not properly designed, documented or tested. These are the things that could be exposed by development, which introduces new bugs and requires additional efforts to fix, when you build the software.

So basically, that's what technical debt is all about. Yeah, so to start with, I would like you to share with me your name, the course you're doing, the year you're studying, and the role you're performing and the group that you're working with. Are you available now?

Yes. So, my name is Rans. I'm doing computer science. I'm a journalist. And the group that I'm working with is an online blockchain system, which is based on Ethereum blockchain.

Ethereum brings the software development to the people. Okay, thank you so much for that intro. So, I just wanted to ask you, come again, you said that the group you're working with is basically on a mobile app?

No, it's a web application. Web application. Yeah. And you said you had to use it for different functions? It's different in that it's the name of the project, or its purpose? Its purpose, yeah, basically. The purpose is to conduct a pilot learning blockchain system, which is going to reduce the number of, actually, it's going to reduce fraud.

It's also going to bring in transparency in the voting system. And also, yeah, it's going to reduce those issues of not serving, not serving the voters. So, what are the objectives or requirements of your software prototype? What are the requirements you would consider?

Requirements? Yes. This card, yeah? How many users are doing it? Both. Yeah, so the other requirements, let me see, 8 gigabytes of RAM, or 4 gigabytes of RAM. It will reduce the number of Linux.

The stable Linux. How about the software part of it, maybe the stakeholders? Oh, so the stakeholders, I think, are going to be any person within Uganda who is above 18 who is reasonable to vote.

So, I'm guessing from that. Okay, so thank you so much for that. When the interview is going to be, it's categorised in three, in four themes. The theme number one is about technical debt identification. I hope you remember what we talked about, what technical debt is.

Yeah, so the theme number one will be technical debt identification. Theme number two will be technical debt measurement. Theme number three will be about technical debt impact evaluation. Then last thing we shall look at, are the data improvements. So, going to technical debt identification, so in your experience as a software developer, what are some of the stages that you will also develop in the software prototype?

Developing a software prototype? Yes. Okay, so the first step is performance monitoring. Yeah. The other one is performance. And what are the review meetings? Review meetings, it's been number six to review and assess and also validate the performance of the unit.

The review meetings are done by our software team, like the team that work with all the stakeholders or all the users of the system. For some review meetings, the users of the system. In this example, all the users of the system have been reduced by the requirements of the unit. Oh, that's awesome.

So, after the requirements, you do the validation of the requirements, then you go, you do the designing or you do also like, you go straight to implementation? No, like after validating the requirements, then you proceed to the designs. Okay. After the design phase, we came up with the boards, you know, CD80 and the ISD80 mock-up.

Hmm. Then after getting the mock-ups, we again validate the mock-ups. So, we have preliminary mock-ups. Wow. Then we are inlining the requirements. After that validation stage, then you proceed to assign rules. Assigning rules in boards.

It's going to work for each part and it's going to work for each part. After assigning rules, then you proceed to implementation. We name it, but we'll use this part. After implementation, we use a name, then we give it a statistic. Then we get the results and changes. Okay, that's awesome.

You are doing some good work there. So, what are some of the key challenges that you normally encounter when you are using the tools? You normally use all the standards, the framework, to look at the programming languages and other conditions that you normally use when you are doing this. Oh.

So, tracking of the test experience in Trello board. Yes. Trello board, we are using tracking in German for years. We use it also in ES6. And designing is a big one. Where do you go to work with it?

I'm talking about maybe the challenges you normally face when you are using these tools. Oh. Yeah. Some of them are primitive version because we need the first people. They have been tracking for 40 nights. Then Figma is a bigger version which has a lot of functionality compared to the few ones.

So, that's one of the challenges. And also, these tools, they need some video experience to be able to use for someone to work. Yes. Yes. So, how do you normally identify technical gaps in your project? What do you mean by technical gaps?

I was explaining about technical gaps. Normally, technical gaps is the same as technical debt. It's used interchangeably. I could ask how do you normally identify or how do you normally become aware of technical gaps in your project? Can I say a question?

Yes. I'm trying to be very brief in my language. You know, when you are doing the projects, when you say you've told me that you have those review meetings, that could be one of the ways you normally sit down and you say, when we implement this shortcut, it's going to affect our project or it affected our project in this way or the other.

Or when we collected the requirements, they were not enough, so we have to go back. So, those could be one of those ways that you normally consider when you identify those technical gaps in your project. So, I could interrupt you, but for you, which one do you normally use to identify those technical gaps? Hello.

Can you hear me? Yes, I can hear you now. You got my question? No, I have some little issues. Yes, so I'm trying to, you know, like you told me, after each stage of software development, you do what you call review meetings.

So, that could be one of the ways when you are evaluating requirements with the stakeholders, then they tell you, they've been in the dialogue, but you have not created a dialogue in the project. So, that is also a technical gap. Maybe you guys took a shortcut of just using in-service data instead of collecting real data, and it has affected your project, then the software that tells you, you go back and collect more data.

So, you are taking a lot of time in collecting more data, so that's also a technical gap, because it's incurring a lot of time on you, and then it's reducing your productivity instead of implementing new features. You are collecting data that you've done when you collected, maybe you've performed the right procedures. So, I'm asking, how do you normally identify technical gaps in your project?

Yes, basically, how would I normally identify technical gaps? Yes. By comparing the time we've been working. Okay. Yes. That's one basic review. So, you are about to begin with the review meetings, and there are some stages where you reach and you identify the gaps?

Yes, I remember at every stage we come up, at every stage we do review meetings. So, when we do a stage of implementation, we do the implementation, then we review, then we immediately get time, we manage the corresponding requirements. If there is one of the requirements is not fulfilled, then we have to go back and do a review meeting,

and then we work on the requirement, which is not changed. Oh, that's wonderful. Thank you so much for that. At least now you've understood this, so that's what I tried to mean by that question. So, what are some of the code metrics or indicators or direct flags in your code,

that may show you that you are leading to a technical gap in your product? The metrics I'm talking about could be code duplication, the complexity of the code, code changes, frequent code changes, it could be code equality, it could be coverage, the testing and the code ownership where someone owns the code, but the rest of the team doesn't know what the other person is doing.

So, what could be one of those metrics that suggests that you are leading to a technical gap in your product, specifically in the final product you guys are working on? Code changes. Code changes, they are so frequent. Yeah, so, how do you normally identify, how do you use, like a control version?

Control? Like a git, value control tools. I was like, do you normally use value control tools like git? Yeah, yeah, yeah, we use git. Oh, that's wonderful, that's one of the tools that normally will help you to identify that code changes, because you are going to keep track of who has done that, who has done what.

Yeah, so, in team number three, we are looking at technical bit measurement. You can, you can demonstrate now what a technical gap is, right? Yeah. Yeah, so, how would you like to measure a technical gap in your product? How would you like to measure it?

Like, if you have identified it, like we've seen, like maybe you've, you told me, you look at code, for you guys do the review meetings, where you contact the stakeholders, so how would you like to measure those technical gaps? Maybe if you've identified the code duplicity state, or the code complexity, in your code, like for you I've said that you have talked about code changes,

so how would you like to measure it? I think the best way, the best way is to, to identify it, identify it with the, the corresponding requirements, because if the requirement is that the user is supposed to know, you know, an identity, an email, and for you, you've given the details,

that the user is supposed to know, you give them email and the password, then, then that one is not valid. That's not, it's not a valid requirement, because the requirement is telling us to know the image of the key and the image of the key. So that's the best way to assess. Okay, so, which one of the current tools that are available for you to normally measure technical gaps in your, in your project?

Yeah, I mean, it's a little bit difficult, I find, to keep track of the changes, right? Yeah. Yeah, so have you heard of tools such as tape size, sonar cubes and code climate? No. So, these are one of the tools that you normally use to identify technical gaps,

or to measure it in your project. You can just install it as an extension in your, in your video studio code. So it brings for you graphs and identifies specifically, and put them in your, in your code. Then you've written a long line of code, and then it tells you, by writing the same line of code,

after some time you won't understand it, so it will alert you. Even for you, you talked about code changes. So when you use, when you use code climate or sonar cubes, what it will do for you, it will identify that someone changed this line, and you're also changing it. So it will lead into a magical flip, rather than spending a lot of time

trying to figure out, trying to, to resolve this magical flip. So it alerts you, so by the time you are changing it, you are aware that, no, I'm not supposed to change this, so it's saving you time, not leading into a technical gap. So those are some of the little things that may be a little weird, but after we see the results, now you are aware of them, right?

so let's go to another part that says, how you deal with like, where a provider is like, they're taking your data to your code base, like, which gap should you address first? How do I know?

How you deal with priorities. Which technical gaps to address first in your prototype?

How do I address first? Yeah, like how do you deal with priorities, where to address first? How do I prioritize, what? Technical gap to address first. This is not, maybe like identifying like a technical gap in requirements,

like identifying technical gap in design, like identifying technical gap in, in requirement, like when you say in implementation, in testing, so which one do I prioritize to, to, okay, so that's wonderful. So when you look at that, how do you know,

like how does, like how do you resolve software gaps? Is technical gap, in terms of reliability, performance and maintenance? How does, how does, how do you resolve software gaps, like the technical gaps we are talking about, affect your software product,

in terms of reliability, performance and maintainability? Yeah, how do they affect your product? They don't get the, the performance, they don't get the performance of the project. Yeah, okay. So like the, the morale, like the, the productivity of the,

of the, but how about the productivity itself? How about the quality of the product? Yes. I think it's not a good system, but it's a product. Oh, okay, that's wonderful. Compared to what you guys maybe had estimated at the start. So, can you provide a specific example,

how did the technical gap affected your project? Yes? Like, my first time I was at payroll, like what I saw critical about it, I mean, it was because it's teamwork. Yeah, like, like, when you take shortcuts, let me give you an example. When you take a shortcut,

let me say you go to code from ChatGPt You implement it in your project, but after like three months, you can't even tell what that code is, like, when you're factoring your code. The only solution you have is just remove that code, and implement a new code. That's the way the technical gap, because you're incurring a lot of costs, like to, like, costs in terms of time,

in terms of resources, something like that. That's what we call a technical gap. Yeah, so I'm just asking, can you provide a specific example of how these shortcuts may have compromised your project, somehow or somewhere? Yeah. Yeah, basically,

This is a good example. When, when you're trying to, like, when there's some style you can, yeah, now, yeah, you're trying to go back to the body, which you wrote somewhere back, but when you're back, you don't understand the code. Meaning that you need to implement a new feature

using that code, when you need to add an example. That code. When you first understand the code, which is, which is not, which is not a top-down, like, end-up, you're writing the entire code. Yeah, so. Yeah, which is, which is then consuming. Yeah, which is exactly what the technical gap is. So, let's go to Ion Payment.

You know, once you take out a financial debt, you go to a bank and you get money, you have to always repay, and you always repay with interest, right? Yeah. Yeah, so, are there any practices or strategies in place to encourage Ion Payment or fixing the technical gaps in your development processes? Are there?

Yeah, are there any strategies or practices in place that you know of that may encourage Ion Payment or fixing the technical gaps? Yeah, performing code reviews on a daily basis. Yes, code review, yes. So, you know how to do code reviews, right? Yeah.

Yeah, like, from the college or from the supervisor? From the supervisor. Yeah, that's wonderful. So, there are also some other, some other practices or strategies that may be aware of. There is education awareness, incorporating technical tips in projects,

setting up repayment jobs, continuous improvement of technical tips, incentives and recognition, and then regular code reviews that you just mentioned. So, for you, you think the one that is in place is regular code reviews? Yeah. Yeah, absolutely, that's wonderful. So, what, of those things that we've mentioned above,

yes, what, what, what, which of them will motivate the workers to reduce these technical gaps as high as possible? As high as possible? Yeah, I think it's the code review, because the code review is a usual thing, you know, usually done by,

usually done by the team, the team leads. So, there is pressure on the developers. You make sure that they understand their code. Before, before their code is, code is, you need to explain your code as well. Okay, so, maybe lastly, in this framework, in your framework area, this research is aiming that is to come up with a set of guidelines

that normally, fellow students or young teams or young leaders can follow in their work in their prototypes. So, like, when they take shortcuts, let me say, if they use, they have code duplication in their code, they have code conflicts, they have long lines of code,

Maybe when you get rid of testing, these guidelines will tell them that even though you've already tested your product, in the long run, you will encash such technical gaps. Let me give you an example. If, maybe your project was so good and it needs to be funded by an investor, but when the investors have their experts in software development,

they identify that you never did testing, you never did the code reviews, and your code is too hard to build, to add in a new feature, you know, so, once they release that, they release that and they go to someone who has a code that is qualitative and it is reliable, the product is maintainable, the product is well performing,

so, if we release that because we don't know what technical debt is, so, what are some of those insights that you normally believe, that you believe that if we add in such a guideline, we will enable students to pay off their technical debt as high as possible? What are those insights or other things that you can consider in this research in your view?

Let me give you an example. Like, for me, an example I can give, if I go as a software developer, I'm aware of a technical debt in mind, maybe I copied code from a chance giving team, but I know that if I don't understand what exactly I copied from internet, it will impact me in a way that,

if I don't know it somewhere, they will ask me to design the same product, because they know I'm going to give it to them, but just because I use the code from internet, I won't be effective enough at my work, but if I took time and know, and I acknowledge that I copied this code, I understood it,

so I will be able to transfer the same knowledge to that place. Yeah, so, I'm just asking in your view, is there anything that we can add to such a framework or such a guideline, that we can give young developers or young software teams? That's a very hard question.

It sounds like a bit of thinking. So, because I'm imagining, when you give some other scenario of a technical debt, imagine if you are taking the content from your first day at the university, or first year in the university, and you are taking the content, you are taking the shortcuts,

to be able to get the coursework and get the marks, the grading, is very important, but at the end, after 1-3 years, you won't be able to understand anything to do the recording, because you are just copying, testing and then passing. When you are entering somewhere, because you did computer science,

you go there, they give you anything to do in computer science, you can't do anything because you are just copying, copying and pasting. But if you understood the code, and you understood the right part of taking shortcuts to a project, then at least you can transfer the knowledge to the working environment, that's enabling you to be effective to the organization.

Yeah, so that's the thing. Maybe if there is anything you can add, but if there is nothing, it's ok, there is no problem. Yeah, so maybe just to thank you for taking the initiative, taking your time to be part of this interview. Yeah, for sure, I will contact you when I am done developing the framework,

I will contact you because the framework has to be very detailed, so you look at it and then you compare with the processes that you have been following, if it works well, then you can always recommend that people can use this framework to identify technical debt, measure it, evaluate it, then page it as high as possible, so I will always be contacting you at the end of the research.

So, thank you so much, Kiza, for taking the time and being part of my interview. Ok, time up, we are out of time, have a wonderful weekend.