**Summarised Transcription**

**Theme 1: Technical Debt Identification**

1. Role in the Software Development Team and Contribution:

- Question: What is your role in the software development team and how does it contribute to the overall project?

- Response: I play a role in the software development team where I'm primarily responsible for software development and coding. My contribution involves writing code, building prototypes, and ensuring that the software aligns with the project's objectives.

2. Description of Software Prototype:

- Question: Can you describe the software prototype you are currently working on, including the type of product (web, mobile, desktop, IoT, or a combination) and its intended functionality?

- Response: Currently, we are working on a web-based software prototype. Its primary functionality is to provide a platform for users to interact with various data and conduct specific tasks related to data management and analysis.

3. Main Objectives and Requirements:

- Question: What are the main objectives and requirements of the software prototype?

- Response: The main objectives of the software prototype are to deliver a user-friendly and functional web-based tool for data management and analysis. The requirements include efficient data processing, user-friendly UI, and security features to protect user data.

4. Software Prototype Development Stages and Challenges:

- Question: In your experience, what are the stages involved in developing software prototypes and what are the key challenges you have encountered related to tools, standards, frameworks, programming languages, and conventions?

- Response:

- The development stages typically involve going from mockups to UI development, focusing on functionality, security, and backend implementation.

- The key challenges we encounter are related to the skill level of our team, limited resources, and taking shortcuts during testing.

5. Identification of Technical Debt:

- Question: How do you identify or become aware of technical debt in your project?

- Response: We identify technical debt through testing, but sometimes we take shortcuts during testing phases, leading to issues in the final product.

6. Indicators of Technical Debt:

- Question: What are the indicators/red flags that suggest that there is technical debt in your processes or product?

- Response: Red flags indicating technical debt include code duplication, code complexity, code ownership, code coverage, and changes. Additionally, delays in product delivery and undocumented features in the code are indicators of technical debt.

**Theme 2: Technical Debt Measurement**

7. Measurement of Technical Gaps:

- Question: How would you like to measure technical gaps in your processes or product?

- Response: We measure technical gaps

- through testing in our staging environment.

- We focus on functionality, duplication, and quality assurance.

8. Current Tools for Measuring Technical Debt:

- Question: Which are the current tools if any or measurements you would use to measure technical debt?

- Response: We currently do not use specific tools for measuring technical debt. However, tools like SonarQube and Code Climate have been suggested for code analysis.

9. Prioritization of Technical Gaps:

- Question: How would you prioritize which technical gaps to address first?

- Response: We prioritize technical gaps

based on their impact on functionality and the user experience.

Critical issues that compromise the system are addressed first.

**Theme 3: Technical Debt Impact Evaluation**

10. Impact of Unresolved Gaps on Quality Attributes:

- Question: How does un-resolved software gaps affect the quality attributes of your software prototype (e.g., reliability, performance, maintainability)?

- Response:

Unresolved gaps in our prototypes can significantly impact quality attributes such as reliability, performance, and maintainability. They can lead to delays in product delivery and a poor user experience.

11. Examples of Technical Gap Impact:

- Question: Can you provide specific examples of how technical gap has affected the project outcomes or the end-user experience?

- Response: We've had instances where unresolved technical gaps forced us to take down the system after a release, resulting in resource wastage and loss of potential clients.

**Theme 4: Early Debt Repayment**

12. Practices for Encouraging Early Repayment:

- Question: Are there any practices or strategies in place to encourage early repayment or fixing of technical gaps during the software prototype development process?

- Response: Initially,

we used to roll out MVPs with known issues and fix them later.

this approach led to severe consequences.

Now, we ensure that critical issues are resolved before the MVP is released.

13. Incentives for Managing and Reducing Gaps:

- Question: What incentives or mechanisms exist to motivate the team to actively manage and reduce these gaps?

- Response:

- We do not have specific incentives in place,

- but we encourage continuous improvement, skill development, and seeking mentorship from experienced professionals.

Conclusion

14. Additional Insights:

- Question: Is there anything else you would like to add or any additional insights you would like to share regarding technical debt in software prototype development? If yes, please share with me.

- Response: It's essential for young software development teams to prioritize education, adhere to standards, and avoid taking shortcuts.

* Capacity building and mentorship play crucial roles in improving development processes.
* Continuous improvement and staying updated with industry standards are key to success in software prototype development.

**FULL TRANSCRIPTION**

Basically, the main goal of the research is to identify metrics that students or young teams use when they are identifying technical debt in their prototypes they are coming up with in the startup. Yes. Yeah, so maybe to bring you up to speed on what technical debt is, technical debt is the consequences that you may suffer when you take shortcuts in implementing your solutions. These technical debts are normally compared to financial debt. The more time you take to resolve it, the more interest you pay. And sometimes it's due to poor design, poor documentation, poor testing, and it ends up slowing you down in development. It kills your morale because every time you are on a project, instead of having a new feature to develop, you are debugging. So that's basically my research. I don't know if I've explained technical debt well.

Interviewee (Ben Okello): Yes, you have explained technical debt.

Interviewer: Yeah, so maybe you introduce yourself, maybe the startup you are working on, and then the name, that's all.

Ben Okello: My name is Ben. Ben Okello. I am a student. I am a graduate. And how do you say it? I am a philosophist. Yes. I run a startup. It's called Digisapp. It's a financial technology solution, a life interest solution.

Interviewer: Okay, so basically you work on web applications and mobile applications.

Ben Okello: Yes, I do work on web and mobile applications. I also work on APIs and I do a lot of work on the full stack of mobile. Oh, full stack of mobile.

Interviewer: So, in this process, this research is majorly surrounding four things. First is technical debt identification, then technical debt measurement, then technical debt impact evaluation, then lastly that is technical debt and repayment. So in your experience, what are those stages that you normally involve yourself when you are developing a software prototype?

Ben Okello: Me? I involve myself in the development of anything technology-related in the startup. I oversee it and I give a go-ahead to what is then pushed into production from the development side into the production environment.

Interviewer: Okay, so could you share with us some of the challenges you normally encounter in these processes?

Ben Okello: The challenges, one of the most challenging technical-wise is scale. Scale because our team is a commitment of the team, aligning goals by now. So we have a lot to deliver but we don't have the capacity to deliver it. The expertise of the team? Yes, the expertise of the team, the skill level of the product.

Interviewer: So you believe we always have technical gaps in our prototypes or our software?

Ben Okello: Yes, we take a lot of shortcuts in our development.

Interviewer: So how do you normally identify these technical gaps in your software? How do we identify them?

Ben Okello: We do testing. Testing? Yes, even our testing, we skip some. We are supposed to have an alpha beta test as a user acceptance test but we basically don't do our alpha test as a beta test. We don't have someone else on the team testing. Basically, the second person testing the product would be the end user. So we do beta and beta user acceptance test.

Interviewer: Okay, so what are those red flags that you normally consider when identifying these gaps in your prototypes?

Ben Okello: Red flags? Red flags, let me talk about maybe code duplication, code complexity, code ownership, code coverage like you talked about in testing, maybe also code changes.

Interviewer: So, now one thing we have is quality assurance. Quality assurance? Yes, because at the end of the day we work with the Vs. The only advantage we have is that most of our work is in the staging environment and even our MVPs are in the staging environment. We don't test in production. A lot happens in the staging environment so we find that's where we clear the duplication, that's where we clear functionality and that's where we do quality assurance. It costs us time of delivery. Our product is delivered late to the user so that sets us back. We take longer to make the product.

Interviewer: So the main challenge that causes most of this is skill level. Skill. Skill level, the team, the number in the team, the shortcuts we are using.

Ben Okello: So we test features and the product, as a whole product is not yet delivered to the end user. By a general public who are supposed to be doing the user acceptance test. So now you find we have a limited skill team. We have to develop a web application. We have to do the mobile backend. We have to do the mobile application itself. We have to do the database structure. The APIs.

Interviewer: The APIs.

Ben Okello: So when you have a small team of two people cutting across. It's really hard work. So you find we take shortcuts for testing purposes to have them repeat shortcuts. This is our challenge. In the long run, we end up with bugs. We end up with undocumented features in the code. Our documentation is there but it's not. What we rely on is project management who is to give us, because in the development process we are documenting what we are doing and what needs to be done next stage. So we rely on these tools. Help us build a basic documentation. But we are not operating in a standard way.

Interviewer: So what measures do you normally use when measuring these gaps?

Ben Okello: When we are measuring these gaps we use a program. You call it documentation. Yes, it's documentation. Basic documentation. Along the process, we use Google Sheets. We use a lot of project management tools. We use Pivotal. We also use Celeron. So we tend to use notes. We used to use Slack. But the challenge is that the messages after 90 days are not there. So you lose track of what you are writing.

Interviewer: So those are some of the gaps that are there. So how about these other tools that normally show you that you have a gap in your code? Maybe like using step size, SonarQube or code climate in IDE's. Those extensions. To show me that there is a space gap in my code.

Ben Okello: Yeah. Or like that there is an extra line that is... No, no, no, not an extra line. But it may be these tools enable you to think... Like let me talk about step size. When you install it in your IDE, maybe you are using Visual Studio Code. What it does is it will show you the code complexity. It will show you the code duplication. The code quality. And it gives you the analysis and tells you maybe this is critical. This is not so severe. So it gives you a way to do what? To measure the quality of your errors. And then you can always fix them.

Interviewer: So those are the tools that I was asking. Maybe

if you use any... I have not used such tools before. But one thing I normally document like when we are doing... When we are writing code. We tend to document step by step. Before we at least a clinical architecture. So that aside, we are not using such tools. We are not engaged with them. However, we look into it and we are seeing. Okay. So you look at step size. You look at SonarQube. Most students use code climate because it is very friendly. But there is also step size. It is more detailed. It conducts the testing. There is code analysis. Then it shows you the graphs of how your code is standing. So there you can understand that there are loopholes in your prototype.

Interviewer: Okay. Yeah. Very good. I get what you are saying about the prototype.

Ben Okello: Yes. The loophole is we analyze the code just ourselves. As we write it. You do like code reviews. You use code reviews. You do code reviews for your faculty? Yes. We review the code ourselves. Everything.

Interviewer: Okay. We are always looking at it and looking at it. We get a future. We do actually two versions of the future. We do security checks mostly. We can test one aspect on one side. Then it goes back to our team. So now we go back and test the system as a possible input.

Ben Okello: So when you are selecting which part to fix, like fixing this part, these gaps in your prototypes, which phase do you normally really consider that these gaps should be fixed first?

Ben Okello: Okay. I think as we are developing, we come up with prototypes. We get the mockups. Adjust them from mockups transferred to code UIs. As the UI is fairly there, then we test functionality before we affect the UI to look like a mockup. So as we test the functionality, we go into the security aspect and functionality. How do we want something to function? Is this what the client wants? Then we go back and look at the backend. Is this what we want to achieve? Once we keep on, we have a bigger sheet. We get to do. So we keep on processing out. Does the code do this? Does it detect this?

Ben Okello: So we have steps we have, but the most important aspect where we put the consideration is on the functionality. That's the implementation part of it? Because once the functionality is good, we have this aspect that we're not going to go out with a product that is not ready to be used by the public. By what is having an error to our standard. So you always want to put out there a product that is on par with that has passed. At the point of testing, that's why we do the quality assurance. The challenge is you will have quality assurance on the way back. Because now the way back is what we're doing mostly. On the mobile device, we have to do the same QA or API to add it. So we document and we're still in the process of developing the right thing. So what is working is what we push there.

Interviewer: So when you have the impact of the technical debt, how do these unresolved bugs in your system affect your prototype? Maybe the quality attributes such as... Delay. It causes a lot of delay. Delay. It causes a lot of delay. Of MVP release. Yes. Because now the team is a bit small. Our MVP is directly our staging environment, production environment. So our MVP is our staging environment whereby we're not looking at user interface beautification. But we're looking at functionality and we move it to post-production. That's all that goes to the production guy. Okay, so in your course, have you guys ever maybe something like to do like you take a shortcut, then you're like this, roll out to this MVP and we get the acceptance and we fix these bugs, etc.? Yes, we did that initially. However, we noticed there were bugs that were actually would compromise the system.

Ben Okello: Yes. So we took down the system and had to redo upgrades to it. So by redoing, you say that you are recalling a lot of resources. Yes, we're buying a lot of time. We're delaying the customers. We're losing potential clients. So it was a repetitive process. And today you're showing a client something and it's not a client telling you, there's this, there's that. So we changed that because of those errors used to get stored. In our production version, there are no bugs. If a feature is available, it's fine. If it's not available, if it has anything pending, we don't push it to the production guy. So all those bugs are not handled at Midlode.

Interviewer: So I was going to ask you, let me say what are the rewards you normally gain from fixing these bugs as early as possible before you push the MVP to the customers?

Ben Okello: The reward we get when we fix these bugs is that we have got a solution. Yes, that's working. That's the first thing. If you've got a solution, then you can replicate it anywhere because you have the solution. That's one. Two, it gives us a flow chain of processes. So we know that this process has to work like this and it's working here. We have a benchmark. So if we're looking at something, it has to meet the standard. So we cannot go and roll out something on the web app that is functioning well and go and start giving the same feature on the Android app and it's not functioning.

Interviewer: So sorting these bugs first before we push out this product helps us in the end of the story. It's pushing out a quality product. That is accepted by the users. That's the end reward. Yes. So what are some of the mechanisms or incentives that you believe young teams can maybe accommodate in their software development processes?

Ben Okello: Let me talk about maybe educational awareness. Maybe they're educated about taking the precautions of taking shortcuts. Maybe setting up repayment rewards that users have fixed their bugs as early as possible. They are given grants, something like that. Maybe continuous improvement, encouraging continuous improvement. Yes, yes. I would say continuous improvement of the product. Continuous keeping up to date with technology in the market. I started development with Java and with apps. It's now almost outdated and obsolete. Yeah, yeah. So they need to... I find that's validated. I studied something that is no longer being used two years down the road. Yeah, yeah. I would have gone for Kotlin. So they need to be aware of... they need capacity building so that they know that these are the industrial standards. These are the way things are done. Let me give you an example. Right now, I jump into an app. I can go with your views here. I have a web app that is working a few things. What I do is I go and I get a web view. You know web view? Yeah, yeah. It replicates that and just puts it inside the app. That is not optimized for an application. It will give the user a bad experience on their mobile device. So the team should always be encouraged to come out improvements in development. Impro

vements in development follow standard procedures. I would say the main goal is seek mentorship from people in the market that are experienced who have done things.

Interviewer: Because that's what has made me work with different people. I've seen most people down to actually have some people testing production. That's a headline. Because the moment the stuff is not working and you have to pull it down. Everything is down. You have to again start debugging and start new things. Some people even don't have databases running for development and production and replication. So do you believe we always have technical gaps in our prototypes or our software? Yes, we take a lot of shortcuts in our development. So how do you normally identify these technical gaps in your software? How do we identify them? We do testing. Testing? Yes, even our testing, we skip some. We are supposed to have an alpha beta test as a user acceptance test but we basically don't do our alpha test as a beta test. We don't have someone else on the team testing. Basically, the second person testing the product would be the end user. So we do beta and beta user acceptance test. Okay, so what are those red flags that you normally consider when identifying these gaps in your prototypes? Red flags? Red flags, let me talk about maybe code duplication, code complexity, code ownership, code coverage like you talked about in testing, maybe also code changes. Yes, so now one thing we have is quality assurance. Quality assurance. Yes, because at the end of the day we work with the Vs. The only advantage we have is that most of our work is in the staging environment and even our MVPs are in the staging environment. We don't test in production. A lot happens in the staging environment so we find that's where we clear the duplication, that's where we clear functionality and that's where we do quality assurance. It costs us time of delivery. Our product is delivered late to the user so that sets us back. We take longer to make the product. So the main challenge that causes most of this is skill level. Skill. Skill level, the team, the number in the team, the shortcuts we are using. So we test features and the product, as a whole product is not yet delivered to the end user. By a general public who are supposed to be doing the user acceptance test. So now you find we have a limited skill team. We have to develop a web application. We have to do a mobile backend. We have to do the mobile itself application. We have to do the database structure. The APIs. The APIs. So when you have a small team of two people cutting across. It's really hard work. So you find we take shortcuts for testing purposes to have them repeat shortcuts. This is our challenge. In the long run we end up with bugs. We end up with undocumented features in the code. Our documentation is there but it's not. What we rely on is project management who is to give us, because in the development process we are documenting what we are doing and what needs to be done next stage. So we rely on these tools. Help us build a basic documentation. But we are not operating in a standard way. So what measures do you normally use when measuring these gaps? When we are measuring these gaps we use a program. You call it documentation. Yes, it's documentation. Basic documentation. Along the process we use Google Sheets. We use a lot of project management tools. We use Pivotal. We also use Celeron. So we tend to use notes. We used to use Slack. But the challenge is that the messages after 90 days are not there. So you lose track of what you are writing. So those are some of the gaps that are there. So how about these other tools that normally show you that you have a gap in your code. Maybe like using step size, SonarQube or code climate in IDE's. Those extensions. To show me that there is a space gap in my code. Yeah. Or like that there is an extra line that is... No, no, no, not extra line. But it may be these tools enable you to think... Like let me talk about step size. When you install it in your IDE, maybe you are using Visual Studio Code. What it does is it will show you the code complexity. It will show you the code duplication. The code quality. And it gives you the analysis and tells you maybe this is critical. This is not so severe. So it gives you a way to do what? To measure the quality of your errors. And then you can always fix them. So those are the tools that I was asking. Maybe if you use any... I have not used such tools before. But one thing I normally document like when we are doing... When we are writing code. We tend to document step by step. Before we at least a clinical architecture. So that aside, we are not using such tools. We are not engaged with them. However, we look into it and we are seeing. Okay. So you look at step size. You look at SonarQube. Most students use code climate because it is very friendly. But there is also step size. It is more detailed. It conducts the testing. There is code analysis. Then it shows you the graphs of how your code is standing. So there you can understand that there are loopholes in your prototype. Interviewer: Okay. Yeah. Very good. I get what you are saying about the prototype. Ben Okello: Yes. The loophole is we analyze the code just ourselves. As we write it. You know like code reviews. You use code reviews. You do code reviews for your faculty? Yes. We review the code ourselves. Everything. Interviewer: Okay. We are always looking at it and looking at it. We get a future. We do actually two versions of the future. We do security checks mostly. We can test one aspect on one side. Then it goes back to our team. So now we go back and test the system as a possible input. Ben Okello: So when you are selecting which part to fix, like fixing this part, these gaps in your prototypes, which phase do you normally really consider that these gaps should be fixed first? Ben Okello: Okay. I think as we are developing, we come up with prototypes. We get the mockups. Adjust them from mockups transferred to code UIs. As the UI is fairly there, then we test functionality before we affect the UI to look like a mockup. So as we test the functionality, we go into the security aspect and functionality. How do we want something to function? Is this what the client wants? Then we go back and look at the backend. Is this what we want to achieve? Once we keep on, we have a bigger sheet. We get to do. So we keep on processing out. Does the code do this? Does it detect this? Ben Okello: So we have steps we have, but the most important aspect where we put the consideration is on the functionality. That's the implementation part of it? Because once the functionality is good, we have this aspect that we're not going to go out with a product that is not ready to be used by the public. By what is having an error to our standard. So you always want to put out there

a product that is on par with that has passed. At the point of testing, that's why we do the quality assurance. The challenge is you will have quality assurance on the way back. Because now the way back is what we're doing mostly. On the mobile device, we have to do the same QA or API to add it. So we document and we're still in the process of developing the right thing. So what is working is what we push there. Interviewer: So when you have the impact of the technical debt, how do these unresolved bugs in your system affect your prototype? Maybe the quality attributes such as... Delay. It causes a lot of delay. Delay. It causes a lot of delay. Of MVP release. Yes. Because now the team is a bit small. Our MVP is directly our staging environment, production environment. So our MVP is our staging environment whereby we're not looking at user interface beautification. But we're looking at functionality and we move it to post-production. That's all that goes to the production guy. Okay, so in your course, have you guys ever maybe something like to do like you take a shortcut, then you're like this, roll out to this MVP and we get the acceptance and we fix these bugs, etc.? Yes, we did that initially. However, we noticed there were bugs that were actually would compromise the system. Ben Okello: Yes. So we took down the system and had to redo upgrades to it. So by redoing, you say that you are recalling a lot of resources. Yes, we're buying a lot of time. We're delaying the customers. We're losing potential clients. So it was a repetitive process. And today you're showing a client something and it's not a client telling you, there's this, there's that. So we changed that because of those errors used to get stored. In our production version, there are no bugs. If a feature is available, it's fine. If it's not available, if it has anything pending, we don't push it to the production guy. So all those bugs are not handled at Midlode. Interviewer: So I was going to ask you, let me say what are the rewards you normally gain from fixing these bugs as early as possible before you push the MVP to the customers? Ben Okello: The reward we get when we fix these bugs is that we have got a solution. Yes, that's working. That's the first thing. If you've got a solution, then you can replicate it anywhere because you have the solution. That's one. Two, it gives us a flow chain of processes. So we know that this process has to work like this and it's working here. We have a benchmark. So if we're looking at something, it has to meet the standard. So we cannot go and roll out something on the web app that is functioning well and go and start giving the same feature on the Android app and it's not functioning. Interviewer: So sorting these bugs first before we push out this product helps us in the end of the story. It's pushing out a quality product. That is accepted by the users. That's the end reward. Yes. So what are some of the mechanisms or incentives that you believe young teams can maybe accommodate in their software development processes? Ben Okello: Let me talk about maybe educational awareness. Maybe they're educated about taking the precautions of taking shortcuts. Maybe setting up repayment rewards that users have fixed their bugs as early as possible. They are given grants, something like that. Maybe continuous improvement, encouraging continuous improvement. Yes, yes. I would say continuous improvement of the product. Continuous keeping up to date with technology in the market. I started development with Java and with apps. It's now almost outdated and obsolete. Yeah, yeah. So they need to be aware of... they need capacity building so that they know that these are the industrial standards. These are the way things are done. Let me give you an example. Right now, I jump into an app. I can go with your views here. I have a web app that is working a few things. What I do is I go and I get a web view. You know web view? Yeah, yeah. It replicates that and just puts it inside the app. That is not optimized for an application. It will give the user a bad experience on their mobile device. So the team should always be encouraged to come out improvements in development. Improvements in development follow standard procedures. I would say the main goal is seek mentorship from people in the market that are experienced who have done things. Interviewer: Because that's what has made me work with different people. I've seen most people down to actually have some people testing production. That's a headline. Because the moment the stuff is not working and you have to pull it down. Everything is down. You have to again start debugging and start new things. Some people even don't have databases running for development and production and replication. So do you believe we always have technical gaps in our prototypes or our software? Yes, we take a lot of shortcuts in our development. So how do you normally identify these technical gaps in your software? How do we identify them? We do testing. Testing? Yes, even our testing, we skip some. We are supposed to have an alpha beta test as a user acceptance test but we basically don't do our alpha test as a beta test. We don't have someone else on the team testing. Basically, the second person testing the product would be the end user. So we do beta and beta user acceptance test. Okay, so what are those red flags that you normally consider when identifying these gaps in your prototypes? Red flags? Red flags, let me talk about maybe code duplication, code complexity, code ownership, code coverage like you talked about in testing, maybe also code changes. Yes, so now one thing we have is quality assurance. Quality assurance. Yes, because at the end of the day we work with the Vs. The only advantage we have is that most of our work is in the staging environment and even our MVPs are in the staging environment. We don't test in production. A lot happens in the staging environment so we find that's where we clear the duplication, that's where we clear functionality and that's where we do quality assurance. It costs us time of delivery. Our product is delivered late to the user so that sets us back. We take longer to make the product. So the main challenge that causes most of this is skill level. Skill. Skill level, the team, the number in the team, the shortcuts we are using. So we test features and the product, as a whole product is not yet delivered to the end user. By a general public who are supposed to be doing the user acceptance test. So now you find we have a limited skill team. We have to develop a web application. We have to do a mobile backend. We have to do the mobile itself application. We have to do the database structure. The APIs. The APIs. So when you have a small team of two people cutting across. It's really hard work. So you find we take shortcuts for testing purposes to have them repeat shortcuts. This is our challenge. In the long run we end up with bugs. We end up with undocumented features in the code. Our documentation is there but it's not.

What we rely on is project management who is to give us, because in the development process we are documenting what we are doing and what needs to be done next stage. So we rely on these tools. Help us build a basic documentation. But we are not operating in a standard way. So what measures do you normally use when measuring these gaps? When we are measuring these gaps we use a program. You call it documentation. Yes, it's documentation. Basic documentation. Along the process we use Google Sheets. We use a lot of project management tools. We use Pivotal. We also use Celeron. So we tend to use notes. We used to use Slack. But the challenge is that the messages after 90 days are not there. So you lose track of what you are writing. So those are some of the gaps that are there. So how about these other tools that normally show you that you have a gap in your code. Maybe like using step size, SonarQube or code climate in IDE's. Those extensions. To show me that there is a space gap in my code. Yeah. Or like that there is an extra line that is... No, no, no, not extra line. But it may be these tools enable you to think... Like let me talk about step size. When you install it in your IDE, maybe you are using Visual Studio Code. What it does is it will show you the code complexity. It will show you the code duplication. The code quality. And it gives you the analysis and tells you maybe this is critical. This is not so severe. So it gives you a way to do what? To measure the quality of your errors. And then you can always fix them. So those are the tools that I was asking. Maybe if you use any... I have not used such tools before. But one thing I normally document like when we are doing... When we are writing code. We tend to document step by step. Before we at least a clinical architecture. So that aside, we are not using such tools. We are not engaged with them. However, we look into it and we are seeing. Okay. So you look at step size. You look at SonarQube. Most students use code climate because it is very friendly. But there is also step size. It is more detailed. It conducts the testing. There is code analysis. Then it shows you the graphs of how your code is standing. So there you can understand that there are loopholes in your prototype. Interviewer: Okay. Yeah. Very good. I get what you are saying about the prototype. Ben Okello: Yes. The loophole is we analyze the code just ourselves. As we write it. You do like code reviews. You use code reviews. You do code reviews for your faculty? Yes. We review the code ourselves. Everything. Interviewer: Okay. We are always looking at it and looking at it. We get a future. We do actually two versions of the future. We do security checks mostly. We can test one aspect on one side. Then it goes back to our team. So now we go back and test the system as a possible input. Ben Okello: So when you are selecting which part to fix, like fixing this part, these gaps in your prototypes, which phase do you normally really consider that these gaps should be fixed first? Ben Okello: Okay. I think as we are developing, we come up with prototypes. We get the mockups. Adjust them from mockups transferred to code UIs. As the UI is fairly there, then we test functionality before we affect the UI to look like a mockup. So as we test the functionality, we go into the security aspect and functionality. How do we want something to function? Is this what the client wants? Then we go back and look at the backend. Is this what we want to achieve? Once we keep on, we have a bigger sheet. We get to do. So we keep on processing out. Does the code do this? Does it detect this? Ben Okello: So we have steps we have, but the most important aspect where we put the consideration is on the functionality. That's the implementation part of it? Because once the functionality is good, we have this aspect that we're not going to go out with a product that is not ready to be used by the public. By what is having an error to our standard. So you always want to put out there a product that is on par with that has passed. At the point of testing, that's why we do the quality assurance. The challenge is you will have quality assurance on the way back. Because now the way back is what we're doing mostly. On the mobile device, we have to do the same QA or API to add it. So we document and we're still in the process of developing the right thing. So what is working is what we push there. Interviewer: So when you have the impact of the technical debt, how do these unresolved bugs in your system affect your prototype? Maybe the quality attributes such as... Delay. It causes a lot of delay. Delay. It causes a lot of delay. Of MVP release. Yes. Because now the team is a bit small. Our MVP is directly our staging environment, production environment. So our MVP is our staging environment whereby we're not looking at user interface beautification. But we're looking at functionality and we move it to post-production. That's all that goes to the production guy. Okay, so in your course, have you guys ever maybe something like to do like you take a shortcut, then you're like this, roll out to this MVP and we get the acceptance and we fix these bugs, etc.? Yes, we did that initially. However, we noticed there were bugs that were actually would compromise the system. Ben Okello: Yes. So we took down the system and had to redo upgrades to it. So by redoing, you say that you are recalling a lot of resources. Yes, we're buying a lot of time. We're delaying the customers. We're losing potential clients. So it was a repetitive process. And today you're showing a client something and it's not a client telling you, there's this, there's that. So we changed that because of those errors used to get stored. In our production version, there are no bugs. If a feature is available, it's fine. If it's not available, if it has anything pending, we don't push it to the production guy. So all those bugs are not handled at Midlode. Interviewer: So I was going to ask you, let me say what are the rewards you normally gain from fixing these bugs as early as possible before you push the MVP to the customers? Ben Okello: The reward we get when we fix these bugs is that we have got a solution. Yes, that's working. That's the first thing. If you've got a solution, then you can replicate it anywhere because you have the solution. That's one. Two, it gives us a flow chain of processes. So we know that this process has to work like this and it's working here. We have a benchmark. So if we're looking at something, it has to meet the standard. So we cannot go and roll out something on the web app that is functioning well and

go and start giving the same feature on the Android app and it's not functioning. Interviewer: So sorting these bugs first before we push out this product helps us in the end of the story. It's pushing out a quality product. That is accepted by the users. That's the end reward. Yes. So what are some of the mechanisms or incentives that you believe young teams can maybe accommodate in their software development processes? Ben Okello: Let me talk about maybe educational awareness. Maybe they're educated about taking the precautions of taking shortcuts. Maybe setting up repayment rewards that users have fixed their bugs as early as possible. They are given grants, something like that. Maybe continuous improvement, encouraging continuous improvement. Yes, yes. I would say continuous improvement of the product. Continuous keeping up to date with technology in the market. I started development with Java and with apps. It's now almost outdated and obsolete. Yeah, yeah. So they need to be aware of... they need capacity building so that they know that these are the industrial standards. These are the way things are done. Let me give you an example. Right now, I jump into an app. I can go with your views here. I have a web app that is working a few things. What I do is I go and I get a web view. You know web view? Yeah, yeah. It replicates that and just puts it inside the app. That is not optimized for an application. It will give the user a bad experience on their mobile device. So the team should always be encouraged to come out improvements in development. Improvements in development follow standard procedures. I would say the main goal is seek mentorship from people in the market that are experienced who have done things. Interviewer: Because that's what has made me work with different people. I've seen most people down to actually have some people testing production. That's a headline. Because the moment the stuff is not working and you have to pull it down. Everything is down. You have to again start debugging and start new things. Some people even don't have databases running for development and production and replication. So do you believe we always have technical gaps in our prototypes or our software? Yes, we take a lot of shortcuts in our development. So how do you normally identify these technical gaps in your software? How do we identify them? We do testing. Testing? Yes, even our testing, we skip some. We are supposed to have an alpha beta test as a user acceptance test but we basically don't do our alpha test as a beta test. We don't have someone else on the team testing. Basically, the second person testing the product would be the end user. So we do beta and beta user acceptance test. Okay, so what are those red flags that you normally consider when identifying these gaps in your prototypes? Red flags? Red flags, let me talk about maybe code duplication, code complexity, code ownership, code coverage like you talked about in testing, maybe also code changes. Yes, so now one thing we have is quality assurance. Quality assurance. Yes, because at the end of the day we work with the Vs. The only advantage we have is that most of our work is in the staging environment and even our MVPs are in the staging environment. We don't test in production. A lot happens in the staging environment so we find that's where we clear the duplication, that's where we clear functionality and that's where we do quality assurance. It costs us time of delivery. Our product is delivered late to the user so that sets us back. We take longer to make the product. So the main challenge that causes most of this is skill level. Skill. Skill level, the team, the number in the team, the shortcuts we are using. So we test features and the product, as a whole product is not yet delivered to the end user. By a general public who are supposed to be doing the user acceptance test. So now you find we have a limited skill team. We have to develop a web application. We have to do a mobile backend. We have to do the mobile itself application. We have to do the database structure. The APIs. The APIs. So when you have a small team of two people cutting across. It's really hard work. So you find we take shortcuts for testing purposes to have them repeat shortcuts. This is our challenge. In the long run we end up with bugs. We end up with undocumented features in the code. Our documentation is there but it's not.