**More Summarized**

Certainly, here are the summarized and concise responses from the student for each themed question:

Theme 1: Technical Debt Identification

1. T1Q1: Stages in software prototyping vary by domain; familiarity with SDLC varies. Challenge: Switching from React.js to Next.js.

2. T1Q2: Technical debt often identified during implementation and user feedback.

3. T1Q3: Indicators: Error tools, learning on the job, and delay in project delivery.

Theme 2: Technical Debt Measurement

4. T2Q1: Uncertain about measurement; software not working considered a measure. Interested in using tools.

5. T2Q2: Current tools: Git (version control), ESLint (code formatter). Haven't used Code Climate, Step Size, or SonarQube.

6. T2Q3: Prioritize gaps using scheduling algorithms, like priority algorithms.

Theme 3: Technical Debt Impact Evaluation

7. T3Q1: Unresolved gaps impact deadlines and prolong development; not using best practices lowers software performance.

8. T3Q2: Example: Eliminated feature due to numerous errors and time constraints. Prioritize documentation for feature implementation.

Theme 4: Early Debt Repayment

9. T4Q1: Strategies: Seek expert guidance, conduct team-building, encourage participation in hackathons.

10. T4Q2: Incentives: Team commitment, financial rewards for bug fixing.

Insights:

- Key insight: Create awareness about technical debt for students to adopt best practices.

These concise responses should make coding and analysis more straightforward. If you have any further requests or questions, please let me know.

**Summarized transcript**

Theme 1: Technical Debt Identification

1. T1Q1: In response to the question about the stages involved in developing software prototypes and challenges related to tools, standards, frameworks, programming languages, and conventions, the student mentioned that the stages vary depending on the project domain. They also noted that familiarity with the Software Development Life Cycle (SDLC) varies among students. Additionally, they shared a challenge they faced when they switched from using React.js to Next.js but found Next.js more beginner-friendly.

2. T1Q2: When asked how they identify technical debt in their projects, the students mentioned that they often discover these gaps during the implementation phase when running the application. They also highlighted the importance of getting feedback from the users of the application as a way to identify issues.

3. T1Q3: In response to the indicators or red flags of technical debt in their processes or product, the students mentioned using error indicators provided by tools as hints of errors. They also described technical debt identification as a learning process that occurs while coding. Interestingly, they indicated a lack of understanding regarding why poor testing might be an indicator of technical debt and emphasized that failure to deliver in time is seen as an indicator.

Theme 2: Technical Debt Measurement

1. T2Q1: Regarding how they would like to measure technical gaps, the students expressed uncertainty about how to measure technical debt in their codebase. They suggested that a measure could be when the software does not work, and they expressed interest in using specific tools for measurement.

2. T2Q2: When asked about the tools they currently use to measure technical debt, the students mentioned using Git for version control and code formatters like ESLint. However, they admitted that they have not used other tools such as Code Climate, Step Size, or SonarQube.

3. T2Q3: In response to how they prioritize which technical gaps to address first, the students mentioned using scheduling algorithms, specifically priority algorithms.

Theme 3: Technical Debt Impact Evaluation

1. T3Q1: When asked about the impact of unresolved software gaps on the quality attributes of their software prototypes, the students mentioned that these gaps can result in a failure to meet deadlines and can prolong the development process. They also pointed out that not using best practices can lower the performance of the software.

2. T3Q2: In terms of providing specific examples of how technical gaps have affected project outcomes or the end-user experience, the students shared an example where they had to eliminate a project feature due to the high number of errors that needed to be fixed within a time constraint. They also emphasized the importance of prioritizing documentation to ease the implementation of features on the prototype.

Theme 4: Early Debt Repayment

1. T4Q1: Regarding practices or strategies to encourage early repayment or fixing of technical gaps during software prototype development, the students suggested contacting experts for guidance, conducting team-building activities to ensure team efficiency, and encouraging students to participate in competitive software events like hackathons.

2. T4Q2: In terms of incentives or mechanisms to motivate teams to actively manage and reduce technical gaps, the students emphasized the importance of commitment among team members. They also mentioned that financial incentives can serve as motivation for teams to address and fix bugs in their software.

Insights:

The students shared an important insight about creating awareness about technical debt among students so that they can implement best practices right from the start of their projects.

These contextualized responses provide a more comprehensive understanding of how the students perceive and address technical debt in software prototype development. If you need further clarification or have additional questions, please let me know.

Full transcript:

I apologize for any confusion earlier. Here's the first part of the transcript without the themed questions:

Student Response:

Technical debt, or technical gap as I would express, the main objective of this research is to identify metrics of measure in technical debt software prototypes and development processes. So, basically, I may be bringing you today to what technical debt is. Technical debt is more of consequences of taking shortcuts or compromises in software development processes.

Sometimes, it's not really always, it's just a financial debt. It accumulates interest over time, making it hard to maintain and update your software in the future.

It also increases the need for power and challenges that arise when code is not yet designed, documented, tested, ready, communicates to the development. Okay, so, also, the result has, maybe, sometimes, it's not done, the development, the service is in the bag, and you require a little more extra effort in order to reach and improve the quality of your software.

So, basically, what technical debt, in layman's terms, is more of when you take shortcuts, maybe, you use a phone from the internet and don't understand it and implement it in your project.

With time, after some, maybe, three months, four months, you cannot understand what you are doing. So, you have to go back and check through what you are doing and learn what you are doing so that you can use it.

So, the time you are spending, that is also a technical debt, because you are doing something that is a little harder than you get used to. And sometimes, maybe, if you don't document your code properly, after some, after a while, you go back to your code and realize what you are doing.

You try to understand what you are doing, so that is also a technical debt. And sometimes, you do some code changes, maybe, you are on-prem, because of your familiar project, your codebook, but because you took shortcuts and you never used the optimal solution,

you don't know where you are coming from, and because you are using shortcuts, you end up not being productive. So, the time you are dealing with something, you have to go back to your codebook and learn what you are doing.

You cannot keep up with everything, your code is buggy, you cannot deal with the rules, so all those are effects of technical debt. And the aim of framework is to have a guideline for students to be able to repay their technical debt and try to address the technical debt.

Technical debt as the identity of your project. So, that is an overview of what I am going to be talking about.

Maybe, I am going to request you to tell me more about your role in your project, your area of study, and what is your role in your project. Hello, top of the morning to you.

I am of the names of Jamil, Savita and Paul. I am a third year ex-student doing computer science at Uganda Christian University. I hope you can hear me. Yes, I can hear you. Yes, please. So, what is the other thing you said? The loan, the loan in the project. Where? The loan you are paying on the project.

There are quite many projects we are doing. Not the final year project yet. That is something which is to come. But as for now, we are working on it. I won't give you the exact details. I will give you the roles. Yes, just the roles. Because the project is kind of confidential as for now.

So, right now, I am playing the role of the front-end engineer in my group. So, I am basically working in the front-end. What I am using is Next.js. What I am using is Next.js.

Then there is that. So, should I speak about the software? Because we are working on a software project. Then we have a hardware project. Software project, basically. Because I was going to ask you, are you working on a web application, mobile application, IOTO, combination of those?

I am working on a web application. Yes. So, we have just come up with a scope. We are using Trello to manage our projects. So, basically, that is the scope we have.

We are hoping to start on it this week. We have a time frame of about two months. Two months to come up with that. So, we are a group of about four or five people actually. I am a front-end engineer.

We have a back-end. We are working on the UIUS. Yeah. Yeah, that's a great team. You have also maybe DevOps or something like that. People who can do the deployment or you yourself can do the deployment. We are going to do the deployment ourselves.

Okay. So, have you heard about technical data before? Or it's your first time? It's my first time hearing about it. But you realize it's just a term, but the ingredients, these are the things that we are doing, right? Yes, I realize the impact of it actually. What you find actually most of the time is most people go into sites such as, let me say,

who are in search for things they don't know what it is. They struggle to understand these things before they implement them. So, at least you are taking a technical data. But sometimes, let me say you have a startup and you want to move it first.

So, are you understanding technical data at go? Maybe you are like, let me take this loan. Let me buy you as a mate like the one you have in financial data. Yes. Let me take this shortcut instead of documenting, instead of testing, instead of doing analysis.

Let me just come up with an MVP. I give it to my stakeholders. I get their feedback. Then I implement. That one is what we call intended technical data. That one you understand what you are doing and you can always improve.

That one is always good. But the one we are talking about is the one that is intentional. The one you don't know that you are going to accumulate it and you are not able to repay it. And you never know if you will ever repay. But it's always in your product. Some technical data is good because it's going to allow you to get feedback. Then you implement what the users have told you to implement.

Yes. So, maybe in your experience as you are doing your software development, what are some of the stages that you involve yourself in developing the prototype? And what are some of the key challenges that you encounter when using maybe the tools, the frameworks, the programming languages, and all that?

As per now, I don't think I have yet encountered because Next.js is exactly a new thing I have begun using. I used to use React's write.

So, the main issue which used to be with that was basically the routing. Routing whereby you connect the different pages. Can you get to me? Yes. You connect to those different pages so you find that you have an error mostly around there.

So, it takes some good time to debug such a thing. So, it's where you find you are lagging behind a bit. But when you come to Next.js now, this framework really helps you to do the routing.

So, you find that your work is lessened. So, what are some of the stages that you involve yourself when developing

a prototype? What are some of the stages that you involve yourself when you are developing a prototype?

Prototypes, basically. How should I say it? So, when you are coming up with your project, what are some of the stages that you go through? It depends on what you are coming up with. Let me give you an example to explain this.

The same amount of confidentiality that we have been there to develop. Let me say an app for something which is military-based. It's not the same that you will do for someone who connects you to different hotels.

So, you are supposed to go through a research kind of thing. You are supposed to find out from such people who are already in that kind of market. I don't know if I am answering your question. What I meant, you have heard about what we call SDLC, Software Development Lifecycle?

Not really. I am not so familiar with it. But what you are saying is right, because that one is what we call ideation. In Software Development Lifecycle, that one, you start with ideation.

You do the planning, you do the requirements gathering, you do the analysis of the requirements. Then you design, then you come up with the prototype. Then you do what we call the implementation or the coding.

You do the testing, you do the deployment. Then after deployment, you do the maintaining. So, you keep the cycle for the next version. So, that's what we call SDLC. I know these things, you do them without your knowledge. Because you don't know who you are.

So, how would you identify or how would you... Like, when you are doing your... Last time you said you were working with the SDLC and it was giving you some writing errors. So, in your coding, even right now, I know you identify some bugs.

How do you come out of those bugs? So, actually, the bugs which I discover are basically when you are still doing the coding before you deploy or anything. So, the coding works in a way that you can't proceed to the...

I've gotten the key down for it, but you can't proceed to the next thing without this thing working. So, it will have a disciplinary... When you are running the application before you deploy it. By the time we are deploying it, we are sure, or at least 95% sure, that there are no issues.

The other thing that now come up, when, let me say, you've deployed the app, then someone... You're getting feedback from the users. Then someone says, ah, this coloring, whenever I do this, this happens, and I don't like this and all that.

So, you use that feedback to improve that application to do better. Oh, that's great. That's great when you get the feedback from the users, because they are the ones who are going to assist you. Yes. So, what are some of the...

Or things that you have a real technical gap in or... Pardon? What are some of the... What are some of the... Or what are some of the things that you need to keep in mind when you point out that this is a bug, or this is an error in my code?

Anyway, there are different error messages. Some of them are... Some of the programming languages you use, you find they are case sensitive. So, in case, let me say, you have put... Let me use the element of capital letters somewhere.

And small letters. So, there are some languages which accept only small case letters. So, in case you have something that it will show you, you have an error, perhaps, on this line. Oh, basically it uses the in-built tools.

Yes, the in-built tools. You are using VS Code to tell you that you have an error in such and such a line. In such a line. And the error in the file means that it's on that line exactly. Because you may find sometimes that there is an error showing that it's on this line. Yet, what is causing something on that line is something which is a bug.

Like, for example, to give you an example, there are some imports we make. Because some things you can't use. So, in case, let me say, you are using... Let me say, Bookstrap. Yes. Because you don't want to go too much through the design. And you haven't imported that into your code.

And you are just using it in the sense that it will complain that there is a problem with this line. It's actually the only issue that you haven't imported that functionally into your particular code. Yes, yes. I understand. I will say that in content. Yes. So, there are different error messages. Some you can see and understand them.

Some, they keep coming as you move on. So, in case, like I said, some kind of like a learn-on-the-job process whereby you see an error you have not seen before. There are even now options. You either go up, search on the Internet, you see what the list error has been and all that.

Or you go to someone who is, let me call it, much better than you and consult for some help. Yes. And what are the code metrics? Maybe you may not use code, they may be a technical term.

But I know you've talked about, of course, maybe code duplication, code complexity when you have long lines of code. Yes. You've talked about, maybe if you frequently change your code, maybe your code is not ready.

Maybe, let me say, after testing, you've not done enough testing. Maybe your code has a group. Maybe your code is only one person.

It's only one person who knows the front-end, knows the back-end. The more you know about the configuration, the more the rest of the team doesn't know what the code is. So all those that I've mentioned before, the duplication, the complexity, the same with the code, the same thing if you don't do enough testing. Maybe the code is not ready, you are unsure that you really have a technical error.

Maybe also, maybe you've lost the whole code. Maybe you have poor documentation, you have not documented your code. Maybe the supervisor or the team leader has told you that you are having a technical gap in your application.

Maybe also, maybe you are adding a feature to your product. Maybe you are adding a feature like something to your product. Maybe you are adding a feature to your product.

That means you have a technical gap, right? Yes. So you realize all those things that I've mentioned about there are indicators of a technical gap in your project. Yes, there are indicators though. I've not understood, really, how testing is one of them.

Certainly, I'll regenerate the second part of the interview transcript. Here it is:

Interviewer: Yeah, so in your experience as you were developing software, what are some of the software development stages that you go through and the key challenges that you encounter may be related to the standards, frameworks, and programming languages that you use?

Student: Some of the steps I normally go through is the ideation phase, trying to come up with an idea for the project. Then the next one after that, we normally have, after the ideation, ideation normally takes a little bit of a while. But should I also stick to the problems I face in ideation as well? If you can.

Interviewer: Hello?

Student: Yes.

Interviewer: Hello. Hello.

Student: Hello. Can you hear me?

Interviewer: I think it was a network issue.

Student: Yeah, I was asking that. Where did you last hear me from? Like, what was the last thing you had?

Interviewer: Yeah, you're on ideation, like, ideation of the project ID or something like that.

Student: Yeah. So ideation normally isn't that much of an issue. So maybe the problem that comes that they should come to what an ideation is maybe choosing a specific. Yes.

Interviewer: And maybe trying to see whether it's actually a solvable problem, because there could be a situation where I end up picking up a project that seems good on paper, but when the problem is not solvable, it's out of my, let's say my skill level.

Student: Yeah. Or maybe I just don't have the tools to solve that problem. So you say that's one of the problems.

Interviewer: The ideation is the first step, then how the next steps do you go about them?

Student: Then the next step would be trying to design a prototype. So from actually, after ideation, so within ideation, I when I pick a specific problem to solve, I try to see how I can solve that problem. Then if I find a way a specific way to solve that problem, then I go to the point of building a prototype to see if there's I can try to build out something or maybe just have a simple structure of what the solution I want to build could look like.

Interviewer: Okay. So what are those key challenges that you encounter using maybe if you're using a framework or standards or tools?

Student: Well, personally, I haven't been using any specific frameworks or tools. But because normally, I would do try to do like a simple wireframe of the application, maybe like a paper wireframe.

Interviewer: Yeah, yeah, yeah.

Student: I expect the software application to look like. So yeah, so a simple wireframe, then from that I could move to a digital wireframe on something like Sigma.

Interviewer: Okay, then I tried to build out a prototype from that. Great. Yeah, I think the rest now comes in after the prototype, I think there you go now, design something like that, or you straight go for the implementation?

Student: No, no, no, there'll be obviously a little bit of design involved. I tried to plan out and see what tools I would be going to use, what would be the best option, maybe software wise or framework wise for the problem I'm trying to solve. Then from there, that's where I would go to get something out as quickly as possible.

Interviewer: Yeah, so that you come up with an MVP.

Student: Yes.

Interviewer: Okay, great. So how do you normally identify or how do you become aware of technical data in your project?

Student: Some of the ways where I identify technical data, I think the very first telltale is, should I say, the bugs that show up, which are normally quite difficult to track down. The bugs that show up and like you can see where they're coming from, but then there are others whereby I have to go down like a long list of errors to find the actual cause of the problem. And also another thing that where I can spot technical data from is when I try to change something and I realize that I have to go through quite a number of steps to change that specific part of the application that I've built. That's another way to identify that there's a little bit of technical data involved in that part.

Interviewer: Okay. So do you normally do like code reviews or pair programming, maybe contact stakeholder feedback, something like that?

Student: I've done that. I've done that when I was on my internship. I actually used to do that for almost every feature that I tried to push, we would be assigned a team member to work with. So there'd be normally two people on a project. So I would try to do a review with my, there'd be a supervisor, but also with my team member, try to see if the feature is actually doing what we want it to do. And if possible, also try to look into the implementation of that specific feature before the code is merged. So that's kind of the, should I say review that would be done. And during that review, you could work like identify some bugs, right?

Interviewer: Yes.

Student: Yeah. So maybe another question, actually,

as if, as if you answered it, but let's go to it and see what are some of the indicators or red flags that suggest that there is a technical date in your processes or your product?

Interviewer: That's the end of the second part of the interview transcript. Please let me know if you would like me to continue with the third part or if you have any other specific requests.

Certainly, here is the correct third part of the interview transcript:

Interviewer: Okay, so this is the most important part of this framework, and that is early repayment. Remember when we say early repayment of the technical debt, remember when we say that in a financial institution, when you borrow money, it comes up with a new price. So, if you borrow the money today and you pay it tomorrow, it is not that you will not suffer the same interest as someone who borrows the money today and paid in one month, right?

Student: Yes, please.

Interviewer: So, the same applies to you. Someone may say, maybe that I paid in one month, let me fix it now. You will not suffer the same consequences as someone who will take years to fix that credit card. Yes, please. The second question could be, are there some strategies that can carry the repayment or fixing of the debt during the time of your involvement?

Student: Yes, there are some of them. One of them is, let me call it an easy go-through.

Interviewer: Where am I? In case I approach you and I tell you, sorry the name is hard for me to read, let me use Mugoya. I came and tell you Mugoya, I have come to you with a software developer. I want an application that does A, B, C, D and all that. We discuss all the terms and it is what we get. I am giving you the most specific criteria. It has to do with Next.js. It has to do with all the different software, the different frameworks I am giving you. The first and easy go-through that I will give you is, because you do not want to work alone on this. Unless you are that part of the loaner. So the first thing, I am just correcting you, if you want to work with, consider these frameworks here. However, you know that this person is actually so good at this. So let me call it team building. The team you are building to start on this project has strengths in all those areas that you are actually speaking about. There are strengths in all those areas that you find that it is easier to, in case you encounter an error, let me say, on this side. There is someone who is actually good at that, who can easily solve that bug faster than you could have used, who is a bit shallow in that area. That is the one where you can require efficiency.

Student: Wow, that is great. So you basically do the team building. The team building is a very important aspect. You don't give someone back end when they are good in front end and do the vice versa. So you find that each and every one is comfortable. You find that when you come to university, there are so many languages you go through. But you find that each and every one is good in a particular language. So that is one of the main things I will give you, team building. Then, I am forgetting the second one. I will call the question again. Our question was saying, are there any practices or strategies in place to encourage our students to fail their studies as early as possible?

Interviewer: Yes. Then the other thing is, also, there is something called the hackathons. The hackathons, yes. Whereby, you go and you compete. Usually in that, I decode it hackathon. Actually, I typically term it the simplest terms as practice. If you find that I am in practice, usually you practice using that a lot. You will find that I will encounter these errors more. So the more I encounter them earlier, before I actually work on a real project, the better for me. You will find that when that time really comes, I am actually, like you have seen before, if I see this, I know this is where the issue is, this is where the issue is. I will find I am more efficient. Are there any ways I can give you to solve that debt? Because you have already encountered them in the world. I have already encountered them. That's it. That's awesome. There are two I can give you.

Interviewer: So, what are some of the incentives or mechanisms that exist? You are breaking it. I will repeat the question. You have been breaking it. Let me come again. What are incentives or mechanisms that exist to motivate the team to actively manage and use these valuable tools and apps in the software?

Student: It's basically just commitment. There is nothing else I can give you about that. If there is no commitment, then there will be a time lag. For example, let's meet tomorrow and we are doing this. If someone doesn't show up, that means someone is not committed to the project. It's basically a laugh. That's why you find that some people, if we all join at the same time in the first year, during the way, you find for a branch not to go to different parts because they don't have that laugh. But if someone is committed, that's how they push to the end. So, I would say it's basically just commitment. If a team is not committed, then you will have those debts.

Interviewer: That's awesome. So, you basically admit that it's commitment that can push people to solve their errors. For me, it's commitment. The only thing I will give you is a stupid reason. I will say money. Finances, right? Finances. If you are doing this project in this time and you are getting some amount of money, that motivation will also come up and say, let me do these things because I don't. I will miss these future things. Although it's a minor one. It's a minor one, but it's so important. You know when you are doing a project and you are just saying things and a project is going to fetch you some money, you really do pull out your best effort that you can. Yeah, when you put your best effort, then let's get this part. The last part that you look at, is there anything else that you would suggest? Is there anything else that you would add in such a framework to enable students to identify, measure, and then evaluate their technical debt and fix it as well as possible?

**Student: I think it's just the awareness. Luckily, some of the things from this meeting, I picked some few things here. I didn't know about that measuring part of it. I've gotten some of the things like the climate, the step-up size, then the solar cubes. Some of the things I didn't know about them, you see, so you have given me insight about them. So, I'll actually begin implementing it. So, first of all, creating this awareness before you actually begin doing anything. Because remember, before you build anything, first of all, again according to the SDLC, first of all, you first go and consult about it. It's a hard side here to discuss, which I think will help people this way. Then if you are willing to pay for it, then you go ahead. Because basically, the first point is awareness.** Yes. Yeah. Okay, thank you so much. Thank you so much, Peter. You really keep bringing me great insight.