**Summarized codeable data**

**Theme 1: Technical Debt Identification**

- Q4: Stages in project dev: Research, tool selection, scheduling.

- Q5: Identifying tech debt: Online resources, error indicators.

- Q6: Indicators of tech debt: Long code, syntax issues, lack of documentation.

**Theme 2: Technical Debt Measurement**

- Q7: Measuring tech gaps: Use tools like Git version control, code linters.

- Q8: Current tools: Git version control, VS Code with linting.

- Q9: Prioritize gaps: Sequential bug fixing based on component importance.

**Theme 3: Technical Debt Impact Evaluation**

- Q10: Impact of gaps: Non-functional issues, affects user experience.

- Q11: Examples: Unable to provide specific instances.

**Theme 4: Early Debt Repayment**

- Q12: Practices: Working in components, proper roles and tasks.

- Q13: Motivation: Deadline and financial incentives.

**Insight:**

- Q14: Framework insights: Step-by-step guide, learning tools, documentation, role division, realistic scheduling.

Full Transcript:

Interviewer: Hello, good afternoon. My name is Mugoya Dihfahsih. I'm pursuing a master's degree in software engineering at Makerere University. I'm conducting research in software architecture, specifically focusing on software development in student projects. The main objective of this interview is to identify key metrics for measuring technical debt in software prototypes developed by young teams, such as students. Have you ever heard of technical debt before?

Participant: No, I haven't. This is my first time.

Interviewer: To bring you up to date, technical debt refers to the consequences of taking shortcuts or making compromises during software development processes. It's sometimes compared to financial debt, accumulating interest over time, making it difficult to maintain and update the software in the future. It also represents the extra effort required to fix errors resulting from poor design, lack of code documentation, or inadequate testing. Technical debt can introduce bugs and necessitate significant time and resource investments for later fixes. The main goal is to create a framework that enables students to address these technical gaps as early as possible. Now, could you please share your name, the course you're enrolled in, your year of study, and your role in a current project?

Participant: My name is Sarah Nsereko, and I'm studying computer science. I usually serve as the team lead on our projects.

Interviewer: Great! Now that you have some understanding of technical debt, let's dive into your experience as a team lead. What are the key stages you typically go through when developing a prototype, project, or software application?

Participant: We start with research, then identify the necessary tools, and create a schedule.

Interviewer: What challenges do you encounter when using tools, frameworks, or programming languages during the development process?

Participant: One major challenge is the learning curve associated with certain programming languages. Some team members may excel with one language but struggle with another, which can slow us down.

Interviewer: How do you usually identify errors in your projects or software?

Participant: We often refer to online resources for explanations when errors occur. Indicators of errors may include long lines of code and syntax issues.

Interviewer: Do you employ version control for your projects?

Participant: Yes, we do use version control, primarily Git.

Interviewer: What about documentation? Do you document your code?

Participant: Unfortunately, we rarely document our code.

Interviewer: How would you like to measure technical gaps in your products or projects?

Participant: It would be helpful to use tools that can assess code complexity and identify long lines of code.

Interviewer: How would you prioritize which technical gaps to address first in your code?

Participant: We usually address errors sequentially, starting from the first line indicated by the compiler.

Interviewer: How do unresolved gaps affect the quality of your product or software?

Participant: Unresolved gaps indicate that something isn't functioning correctly, which can negatively impact the user experience.

Interviewer: Could you provide an example of how bugs or errors have affected your project outcomes or end-user experience in the past?

Participant: Unfortunately, I can't recall a specific example right now.

Interviewer: No problem. Let's discuss early repayment of technical debt. Are there any practices or strategies that encourage you to address technical gaps early?

Participant: Yes, we work on individual components, fixing errors in one before moving on to the next. This approach is driven by education and awareness.

Interviewer: What motivates you and your team to fix bugs or reduce technical gaps as soon as possible?

Participant: Deadlines and financial incentives are strong motivators for us.

Interviewer: In the framework we are developing to guide students and young software development teams, what insights or elements would you like to see included?

Participant: The framework should provide a step-by-step guide, introduce learning resources, emphasize proper code documentation, define roles and tasks clearly, and establish a realistic project schedule.

Interviewer: Thank you, Sarah, for your time and participation in this interview. We hope you've gained valuable insights into technical debt and how to minimize it in your programming.

Participant: Yes, I have. Once the guidelines are ready, feel free to reach out to me. You can obtain my contact information from Conrad.

Interviewer: Great! Is there anyone you know who might be interested in participating in this survey?

Participant: Yes, I can recommend someone.