Coding summary

**Theme 1: Technical Debt Identification**

Question 4: Stages and Challenges

Response: Involve all stages. Key challenge: Choosing the right tech/tools.

Question 5: Identifying Technical Debt

Response: Use tests and look for code complexity, ownership, and poor documentation.

Question 6: Indicators/Red Flags

Response: Indicators include code duplication, low test coverage, and single-person code ownership.

**Theme 2: Technical Debt Measurement**

Question 7: Measuring Technical Gaps

Response: Prefer quantifiable metrics (percentages) for prioritization, haven't used specific tools.

Question 8: Tools for Measurement

Response: Yet to use tools, considering SonarQube and Code Climate.

Question 9: Prioritization

Response: Feature-by-feature approach for thorough debt resolution.

**Theme 3: Technical Debt Impact Evaluation**

Question 10: Impact on Quality Attributes

Response: Unresolved gaps affect reliability, performance, and maintainability.

Question 11: Specific Examples

Response: Rushed project led to image loading issues, impacting user experience.

**Theme 4: Early Debt Repayment**

Question 12: Encouraging Early Repayment

Response: Documentation, continuous improvement, and mentorship are crucial.

Question 13: Incentives/Motivations

Response: Organizational structure, clear objectives, and mentorship motivate debt management.

**Insight:**

Question 14: Additional Insights

Response: Clear guidelines on technology choices can prevent tech-related debt.

Summary Transcript

Theme 1: Technical Debt Identification

Question 4: 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: In my experience, I involve myself in all stages of software development, from design to implementation. Some key challenges I've encountered are related to choosing the right technologies and tools. There are so many options available, and picking the right ones can be challenging, often requiring in-depth research.

Question 5: How do you identify or become aware of technical debt in your project?

Response: I use various techniques to identify technical debt in my projects. One common technique is writing tests for my code, especially for specific endpoints or features. Running these tests helps me uncover issues. Additionally, I look at code complexity, code ownership, and documentation as indicators of potential technical debt.

Question 6: What are the indicators/red flags that suggest that there is technical debt in your processes or product?

Response: Indicators of technical debt in my processes or products include code duplication, poor code quality, infrequent code changes, low test coverage, single-person code ownership, and inadequate documentation.

Theme 2: Technical Debt Measurement

Question 7: How would you like to measure technical gaps in your processes or product?

Response: I would like to measure technical gaps through quantifiable metrics, such as percentages, to prioritize which elements to address first. However, I haven't used specific tools for measurement yet.

Question 8: Which are the current tools if any or measurements you would use to measure technical debt?

Response: I haven't used specific tools for measuring technical debt so far. However, there are tools like SonarQube and Code Climate available, which I intend to explore in the future.

Question 9: How would you prioritize which technical gaps to address first?

Response: I prefer a feature-by-feature approach, where I address one feature completely before moving on to the next one. This approach helps ensure that I thoroughly address technical debt in each feature before progressing.

Theme 3: Technical Debt Impact Evaluation

Question 10: How does unresolved software gaps affect the quality attributes of your software prototype (e.g., reliability, performance, maintainability)?

Response: Unresolved software gaps can significantly impact the quality attributes of software prototypes, including reliability, performance, and maintainability. For instance, not addressing bugs can lead to unreliable software and poor user experiences. Moreover, it can hinder maintainability and make adding new features challenging.

Question 11: Can you provide specific examples of how technical gap has affected the project outcomes or the end-user experience?

Response: An example of how technical debt affected a project was when I rushed to complete it, resulting in occasional image loading issues. This negatively impacted the end-user experience as images would sometimes not load correctly.

Theme 4: Early Debt Repayment

Question 12: Are there any practices or strategies in place to encourage early repayment or fixing of technical gaps during the software prototype development process?

Response: Documentation and continuous improvement are essential practices to encourage early repayment of technical debt. Clear guidelines and mentorship, especially during code reviews, can also motivate early debt repayment.

Question 13: What incentives or mechanisms exist to motivate the team to actively manage and reduce these gaps?

Response: Organizational structure, clear objectives, and mentorship can motivate teams to actively manage and reduce technical debt. Supervisors and team leaders play a crucial role in providing guidance and ensuring that technical debt is addressed promptly.

Insight:

Question 14: 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: I believe that providing clear guidelines and recommendations on technology choices for specific tasks or platforms could be valuable. This would help students and developers make informed decisions and avoid technical debt related to technology choices.

**Full transcript**

Hello, can you hear me? Hello. Yes. Oh, yeah. Okay. Maybe this is touchy because something just went along. Maybe you start like you are introducing yourself, your name.

Yeah, sure. My name is Isaac. I'm a computer science student, studying for a year. And your project?

I'm working on different software prototypes.

Okay, right. So, maybe to recap what technical debt is so that you can go deep. These are consequences of taking shortcuts that compromise all that makes your software too hard to be refactored, or when you are adding something, it takes a lot of time and it slows down your development, and it introduces bugs in your software. And it's due to poor design, poor documentation, poor coding practices, testing, all those things. They are the ones that lead to technical debt. So, I would like to start by telling you that we are going to have this interview or a survey in four themes. First of all, we're going to have technical debt identification. Second, we are going to do technical debt measurement. Then technical debt impact, and then the last part would be early debt repayment.

Yeah, sounds good.

Yeah, so in the first part, you know, I do the coding part. Yeah, after pitching here and there.

Okay, so you're in the coding part.

Yeah. So, in your experience, as you're developing different prototypes or products, what are some of the stages that you involve yourself in when you're developing the products?

I'd say, during the development, because as a project manager, I'm trying my best to be involved in each and every single stage, so being on designs and my framework.

Okay, so you involve yourself at all the stages of software development, trying to keep myself in the know of whatever is happening at each given stage. All right.

So, what are some of the challenges that you normally encounter when using maybe, can you hear me? I said, can you hear me? Yes, I can. Yeah, I'm saying what are some challenges that you normally encounter that are related to tools, standards, frameworks, and programming languages that you use when you're coming up with a product?

The challenge would be picking the right one or quote-unquote, right. And actually, you come to realize that there's so many options. So, the challenge would be finding which one is the right one.

So, how do you normally identify the bugs? I must ask that question again, try to rephrase it for me. Yeah, how do you know which techniques do you normally use to identify bugs?

One technique is writing tests for your code. So, what I have done before is write out each and every single endpoint to do with that particular entity, so let's say your entity in this case is users. So, I write every endpoint that is to do with users. Then I write every test that deals with users, then you run those as well. But then yes, it is.

What are some of the indicators, or in red flags, I mean, the code metrics that suggest that there is a technical gap in your product?

I look at code complexity. It's one of them, writing long lines of code.

How would you like to measure technical gaps in your processes or product? How do you measure them?

I prefer quantifiable metrics (percentages) for prioritization. I haven't used specific tools.

Question 8: Tools for Measurement

Response: Yet to use tools, considering SonarQube and Code Climate.

Response: Feature-by-feature approach for thorough debt resolution.

Response: Unresolved gaps affect reliability, performance, and maintainability.

Response: Rushed project led to image loading issues, impacting user experience.

How do these unresolved gaps or bugs affect the quality of the attributes of your software in terms of reliability, performance, and maintainability?

Unresolved bugs well, usually, I don't have unresolved ones. But I'm going to give you an instance when it did because actually, some project that I did a while back...

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

I would say documentation is a good one. Maybe during code reviews, mentors can push you to refactor code.

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

Organizational structure, clear objectives, and mentorship motivate debt management.

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.

Generally, what you're already coming up with is good. One thing that I personally would love would be a really clear guide on like...

So, for you, it's always on the technology side, right?

Yes, that's right.

Yeah. Thanks so much for making your time, Isaiah, and having a part in this research survey.