**Introduction and Research Goals**

Question: Could you please introduce yourself and briefly describe the focus of your research?

Answer: My name is Mugoya Dihfahsih, and I'm pursuing a master's degree in software engineering. My research revolves around identifying metrics for quantifying people in relation to different prototypes. The primary goal is to help students and supervisors minimise software prototypes with disrupted internal qualities or qualities that can't be implemented or built upon in the future.

Question: What is your role, and could you elaborate on the prototype you're working on?

Answer: I'm a Bachelors student in software engineering, currently in my second year at Agarwal Abrams, Computer Science. I'm planning to implement something related to computer analytics, and my role in the project will be on the back end of the team.

Question: How do you plan to support the development of this prototype?

Answer: I anticipate that the prototype will require input data, which will be processed by an engine or model to generate various findings and assess the qualities we're looking for in the system.

Question: Could you explain the concept of technical debt and its relevance to software development?

Answer: technical debt is akin to a cost in software development. It involves taking shortcuts to quickly implement a solution, but these shortcuts can introduce problems that need to be fixed before new features can be added. It's similar to financial costs; the longer you delay fixing these issues, the higher the cost of addressing them later.

Question: Have you encountered technical debt issues in your software development experience?

Answer: I've heard a bit about technical debt but never encountered it directly.

Question: Can you describe the typical procedures you follow when working on a software development project?

Answer: In the software development process, we typically start by identifying the problem, performing necessary engineering or verification tasks, integrating all components, involving users and recommenders, implementing, testing, and finally deploying and maintaining the system.

Question: Have you faced challenges like underestimation or using different languages or tools in your software development projects?

Answer: Yes, underestimation and using inappropriate tools or languages can be challenges. Sometimes the research aspect of a project may be underestimated, leading to problems later.

Question: How can you identify technical gaps or problems in a software development project?

Answer: One approach is to use the research methodology applicable to the project, comparing it with existing systems and technologies. This comparative analysis can reveal gaps or issues.

Question: What are some indicators or warning signs that suggest you might be facing a technical problem in your project?

Answer: An indicator could be when project timelines are consistently not met, or when shortcuts are taken without addressing underlying technical issues.

Question: How would you measure technical gaps during different stages of development?

Answer: Technical gaps should be identified at each development stage using appropriate methods. If the methods being used are inappropriate, the results may not align with expectations.

Question: Are there any tools or techniques you use to identify technical gaps in your software projects?

Answer: Using a research methodology, conducting comparative analysis, and employing techniques like compiler techniques to compare your system with industry standards can help identify technical gaps.

Question: How do technical gaps impact a software product in terms of its functionality, performance, and maintainability?

Answer: Technical gaps can impact every aspect of a software product. If flaws go undetected, it can lead to performance issues, maintenance problems, and affect the overall functionality of the product.

Question: Have you implemented strategies to encourage early resolution of technical debt in your projects?

Answer: In our context, using best practices, such as proper software structuring and version control, helps address technical debt. Failing to use these strategies can result in code loss and other issues.

Question: How important is code documentation in addressing technical gaps, and do you consider it during your software development projects?

Answer: Code documentation is crucial, and I follow the principles of clean code. While clean code should be clear, documentation is still important, especially when working with others or conducting code revision.

Question: Are there any challenges you've noticed in software education related to technical debtand practical experience?

Answer: In our education system, there's often a lack of practical exams and a focus on theoretical learning. This can lead to gaps in practical experience and real-world problem-solving skills, which are crucial in software development.

Question: Thank you for sharing your insights. Is there anything else you'd like to add or contribute?

Answer: Students often do extensive research, but practical aspects and exams are limited. It's important to bridge the gap between theory and practical application in software education.