Participant 3: Mubarak Abdullahi

**Theme 1: Technical Debt Identification**

4. Software Prototype Development Stages and Challenges: <B> discussed stages from requirements to deployment, with challenges related to finding credible information sources, security concerns, and the need for in-depth knowledge.

5. Identification of Technical Debt: <B> mentioned methods include manual testing, comparing expected vs. actual results, and indicators like missed timelines and shortcuts taken.

6. Indicators of Technical Debt: <B> highlighted indicators such as missed project timelines and taking shortcuts without addressing underlying technical issues.

**Theme 2: Technical Debt Measurement**

7. Measurement of Technical Gaps: <B> primarily discussed collective identification of errors and challenges within teams during implementation.

8. Tools for Measuring Technical Debt: <B> mentioned not using specific tools but recognizing their potential benefits, including SonarQube and Step Size.

9. Prioritization of Technical Gaps: <B> emphasized addressing issues during implementation to prevent complexities from accumulating.

**Theme 3: Technical Debt Impact Evaluation**

10. Impact on Quality Attributes: <B> highlighted the impact on reliability, performance, and maintainability, emphasizing the need to identify and address issues.

11. Specific Examples of Impact: <B> mentioned challenges with incorporating data sets from different regions affecting system functionality.

**Theme 4: Early Debt Repayment**

12. Practices for Encouraging Repayment: <B> discussed strategies, including education, rewards, scholarships, and supervisor guidance to raise awareness of the consequences of technical debt.

13. Incentives for Managing Debt: <B> noted the absence of formal incentives but prioritize addressing technical debt during code reviews and refactoring.

**Insights on Technical Debt and Frameworks:**

Participant <B>

* strongly suggests the importance of establishing a framework or guidelines for managing technical debt.
* This structured approach can help identify, prioritize, and integrate debt management into development processes, ultimately enhancing software quality and ensuring long-term project success.

**FULL TRANSCRIBE**

<A>My name is Mugoya Dihfahsih. I'm a student pursuing a master's degree in software engineering. My research primarily focuses on software architecture, particularly in the context of identifying technical debt in student prototypes and other projects that have been developed by young practitioners within a timeframe of less than one year.

The main objective of my research is to identify specific metrics. These metrics include aspects related to software complexity, quality, and other factors that may have arisen during the software development process.

Let me clarify what technical debt entails in a more understandable way. Think of technical debt as analogous to financial debt. When you, as a developer, opt for quick solutions to implement your software in a prototype, it can accumulate over time, much like financial debt that accumulates interest.

These consequences of technical debt can significantly slow down the software development process. You may find yourself needing to invest extra effort to implement new features in your product. Additionally, you might need to revisit and reimplement solutions to better suit your product. In essence, it can feel like starting from scratch, consuming additional resources and time, and potentially impacting other ongoing projects.

<A>To summarize, these are the consequences of taking shortcuts in your software prototype during development. Now that we've discussed technical debt, could you please introduce yourself, mentioning your name, course of study, and the specific focus of your project?

<B>My name is Mubarak, and I'm an undergraduate student majoring in computer science. My research project centers around developing an Explainable AI (Artificial Intelligence) system for diagnosing plant diseases, with a particular focus on maize plants as a use case. This project is ongoing, and I've been working on it diligently.

<A>Now, let's delve into the main objectives of your project and its requirements. Could you explain what the primary objective of your system is and the specific requirements it must meet?

<B>The primary objective of the system I'm designing is to provide accurate diagnosis. Imagine a scenario where a farmer uses an application to capture a picture of a maize plant leaf. Behind the scenes, an AI model classifies whether the leaf is diseased or healthy. The system should not only provide the classification but also offer an explainable aspect, allowing the farmer to understand the reasoning behind the model's predictions. To achieve this, we plan to implement Explainable AI algorithms like Grad-CAM, providing transparency in the decision-making process.

<A>Considering that your model is data-driven, what other requirements, aside from a substantial dataset, does your system necessitate? Are there specific computing resources or tools that play a critical role in your project?

<B>Certainly. Given that we are training our model, it demands a significant amount of image data. Additionally, we require substantial computing power, as we are utilizing a GPU for training. This intensive resource utilization poses challenges, especially due to GPU memory limitations. Consequently, we cannot run consecutive experiments simultaneously, and training one model can take several days. The limitations of GPU resources can be a significant challenge.

<A>Are you working on this project as an individual or as part of a team? If it's a team effort, could you provide some insights into your roles and responsibilities within the team?

<B>This project involves a team of four individuals, including myself. I play a multifaceted role within the team, handling responsibilities related to both the machine learning model and the mobile application. Additionally, I am involved in developing the necessary APIs (Application Programming Interfaces) for our system.

<A>Considering your comprehensive involvement in various aspects of the project, you seem to be a well-rounded student candidate. I understand you've divided your research into four themes. Could you elaborate on these themes, particularly the areas of technical debt identification, measurement, impact assessment, and repayment strategies?

<B>Indeed, I've structured my research around four distinct themes. First, we have "technical debt identification." Next, there's "technical debt measurement," which encompasses evaluating the extent of technical debt in a project. The third theme explores the "impact of technical debt," focusing on understanding how technical debt influences software development. Finally, the fourth theme centers on "repayment of technical debt," where we explore strategies and methodologies to address and mitigate technical debt in software projects.

<A>Now, let's discuss the first theme. In your experience, what were the stages involved in developing prototypes, and what challenges did you encounter during these stages? These challenges could relate to frameworks, standards, tools, programming languages, or other aspects.

<B>Certainly. Starting with the machine learning aspect, we faced multiple stages. Initially, we needed a substantial amount of data, although the lab had collected some maize disease image data. However, cleaning this data proved challenging due to variations in image sizes, formats, and color schemes. Data cleaning and exploratory data analysis were necessary steps. When it came to model development, we experimented with four different models, three of which were pre-trained, and one was custom-made. The challenge here was identifying the most suitable architecture, which required extensive experimentation, taking up to two weeks.

Regarding the mobile application, we utilized React Native, which didn't pose significant issues. TensorFlow Lite, a library for machine learning models, facilitated the conversion of models to a format suitable for mobile apps, optimizing storage space. Firebase was employed to expand the dataset by storing images and their corresponding results. While our experience with Firebase has been relatively smooth, challenges may arise during future integrations as we plan to incorporate additional features.

<A>You've touched upon architecture and experimentation with models. Could you provide more details about the architectural choices made during model development? Did you face challenges due to undocumented aspects?

<B>Certainly. When working with TensorFlow Lite, we did not encounter issues related to the framework itself, but rather in configuring the model's layers. While there was documentation available, it did not provide a specific architecture to use. Thus, we had to experiment with different layer arrangements to optimize model performance. This process required substantial effort.

<A>As you mentioned, your project is a team effort. Given that you have team members specializing in various areas, could you share your perspective on the importance of communication and collaboration among team members to ensure the project's success?

<B>Collaboration and effective communication within the team are paramount for project success. Since we're dealing with multiple components, including the machine learning model, mobile app development, and APIs, seamless coordination is essential. It ensures that all components align with project objectives and that any challenges are addressed promptly. Miscommunication or a lack of collaboration can lead to inconsistencies and inefficiencies, which may impact the overall quality of the project.

<A>In software development, measures like code reviews and refactoring are essential for maintaining code quality. Have these practices been integrated into your project's workflow? If so, how have they contributed to the overall quality of your software?

<B>Yes, we have incorporated code reviews into our project workflow. Before merging any code changes into the main repository, team members review each other's code. This practice has been instrumental in identifying and rectifying issues early in the development process. It ensures that the code adheres to coding standards and is well-structured. Additionally, refactoring plays a crucial role in maintaining code quality. It allows us to improve the codebase continuously, enhancing its readability, maintainability, and performance. Both code reviews and refactoring contribute significantly to the overall quality of our software.

<A>Your mention of code reviews and refactoring highlights the importance of addressing technical debt early. Are there specific strategies or incentives in place within your team to encourage the early repayment of technical debt?

<B>Currently, we do not have formalized strategies or incentives for addressing technical debt early in our project. However, we recognize the value of identifying and addressing technical debt promptly. When we encounter technical debt during code reviews or refactoring, we prioritize addressing it to prevent its accumulation. While there are no formal incentives, the team's shared understanding of the long-term benefits of debt management drives our proactive approach.

<A>Lastly, considering your insights into technical debt and its implications, do you believe that establishing a framework or guidelines could help students and developers manage technical debt effectively in their projects?

<B>Yes, I strongly believe that establishing a framework or guidelines for managing technical debt would be highly beneficial, especially for students and developers who may be less experienced in identifying and addressing technical debt. Such guidelines could provide structured recommendations on how to recognize technical debt, prioritize its repayment, and integrate debt management into the development process. By following these guidelines, students and developers can make more informed decisions, ultimately leading to better software quality and long-term project success.