Lady Linux Transcript: Stakeholder (CS) Interview with Capstone PM

Interview Participants:

- **Project Manager (PM)**
- **Project Stakeholder / Client (CS)**

PM: To begin, can you describe Lady Linux in your own words and explain what problem it is meant to address?

CS:
Lady Linux is a proposed Linux-based operating system that has a language model (LLM) directly built into it. The intention is to use language model conversations to assist with mediated access to the operating system itself, as well as to applications installed on it.

The core problem it addresses is that most people have almost no real understanding of how their systems work or how their data is handled. Even when users are taught basic data literacy or security concepts, the underlying mechanisms remain opaque and are difficult monitor, maintain or update. Lady Linux is meant to bridge that gap.

PM: When you say "data," what exactly do you mean in the context of this system?

CS:
Data is anything the user creates or participates in creating. That includes documents, messages, recordings, images, but it also includes interactions that people don't usually think of as data.

Clicks, scrolls, hovers, session data, cookies, local storage, cloud synchronization, all of these are forms of data creation. Users generate them constantly, but they have almost no visibility into what exists or how it is used.

PM: Why is a language-based interface important here, instead of traditional system settings or dashboards?

CS:
Because the existing interfaces assume expertise that most users don't have. Even technically curious users struggle to understand what settings actually do or how systems interact behind the scenes.

A language-based assistant allows users to ask questions in terms that are meaningful to them. Instead of navigating cryptic menus, they can ask, "What data is this application collecting?" or "What happens if I change this setting?" The system should be able to answer those questions clearly and honestly.

PM: Is the language model intended to make decisions automatically on behalf of the user?

CS:
No. That's a critical distinction. The language model is not meant to be an autonomous agent. It should not make changes without explicit human approval.

Its role is interpretive and instructional. It inspects, explains, proposes, and warns, but it does not act independently. Human-in-the-loop control is non-negotiable.

PM: From a technical perspective, what are the major components students would be working on?

CS:
There are several major pillars. First, the operating system itself, either extending an existing Linux distribution or building from the kernel up. Students should actively decide what utilities, libraries, and services belong in the system rather than inheriting defaults blindly.

Second, the language model. Initially, it should be an off-the-shelf model deployed locally. Over time, it could be fine-tuned using technical documentation, Apache, Git repositories, and system manuals so that it understands operating system internals more deeply.

Third, there must be an abstraction layer between the model and the system. This layer mediates access, enforces permissions, and ensures reversibility of changes.

PM: Can you expand on the abstraction layer and why it's necessary?

CS:
Without abstraction, you either give the model too much power or none at all. The abstraction layer defines *what the model is allowed to see*, *what it can propose*, and *how those proposals are executed*.

It also supports approval workflows and rollback. If a setting is changed, the system should be able to explain the implications and undo the change if necessary. This is essential for safety and trust.

PM: Security is clearly a major concern. How do you see that addressed in the project?

CS:
Security must be fundamental, not an afterthought. There is a growing risk in automated systems that are given excessive authority without safeguards.

Lady Linux must apply least-privilege principles rigorously and be designed so no component, especially the LLM can arbitrarily change the system. Every action must be inspectable, auditable, and reversible.

PM: You've emphasized data literacy several times. How does that manifest in the system design?

CS:
Data literacy is not just documentation, it's interaction. The system should help users understand what data exists, how it flows, and what happens when it's shared or exported.

That means designing representations of data that are comprehensible, not just technically accurate. It also means being explicit about encryption, retention, deletion, and sharing mechanisms.

PM: What role does the user interface play in all of this?

CS:
The interface is arguably the most important component. You can have a perfect system under the hood, but if the interface fails, the system fails.

The UI must be teachable. It should guide users, ask questions, provide explanations, and support learning. A welcome or tutorial component is essential. This isn't cosmetic, it's foundational.

PM: You've spoken critically about modern consumer hardware. How does that factor into the project?

CS:
Modern devices are increasingly difficult to repair or modify, and control PII. This is not accidental, it's a business strategy. Devices are designed to fail in ways that force replacement rather than repair.

Lady Linux stands in contrast to that model. While the initial focus should be desktop and laptop hardware, students should also explore the realities of mobile devices and understand why Linux struggles there. Hardware constraints are part of the system story.

PM: From an academic standpoint, why do you believe this is a strong senior capstone project?

CS:
Each component offers a student an opportunity to engage deeply with real systems, not just code in isolation. It brings together the wide span of computer science topics including operating systems, security, AI, data management, user experience, ethics, and sustainability.

Students don't just build something that "works." They must justify decisions, consider consequences, and communicate trade-offs. That is exactly what a capstone should do.

PM: Finally, what do you see as the long-term value of this project beyond a single semester?

CS:
Lady Linux should not be a one-off. It should be an evolving platform that future cohorts can build on.

The goal is not perfection, it’s direction. If students leave understanding how systems shape human agency, and how intelligent tools can empower rather than obscure, then the project has succeeded.