Date: 12 August 2023

Sarah: Thanks for joining, everyone. Today, our focus is to clarify requirements for our web-based to-do app. Let's start by discussing the core functionality and any initial thoughts you all have on the project.

Bob: Sure thing, Sarah. The goal is to create a straightforward and efficient task management app. It should be user-friendly with key features like task creation, categorization, due dates, and notifications.

Emily: For the frontend, we should prioritize a clean and responsive design. Users will likely access this on both desktop and mobile, so we'll need a flexible layout.

Liam: Agreed. We could use a framework like React to maintain modularity, especially if we'll be expanding features later on.

Raj: On the backend, I suggest using Node.js with Express to ensure scalability. This way, we can handle real-time updates, especially for notifications.

Mia: Adding to that, we'll want to implement a database that can efficiently manage task data. I'd suggest MongoDB since it's flexible for handling different data structures.

Chloe: For UI/UX, simplicity is key. Users should be able to add, edit, and complete tasks with minimal steps. We can incorporate icons for different categories and make the interface as intuitive as possible.

David: From a QA perspective, we'll need to define clear acceptance criteria. I suggest we set up both functional tests for core features and usability tests to ensure a smooth user experience.

Sarah: Great points. Let's also establish a timeline. I'm thinking we aim for an MVP in the next two months, covering core features like task creation, categorization, and notifications. Does that seem feasible?

Bob: Yes, that sounds manageable. We can work in sprints, focusing on one feature at a time to ensure everything is well-tested and integrated.

Emily: I agree. Starting with task creation and basic UI elements will give us a solid foundation.

Sarah: Perfect. Let's document today's decisions, and I'll share a breakdown of tasks for our first sprint. Thanks for the input, everyone.

Date: 18 December 2023

Sarah: Thanks for joining on short notice, everyone. I had a call with some customers, and they suggested adding a feature where users can assign tasks to others within the app. They see it as a useful addition for team collaboration.

Emily: That's an interesting idea! From the frontend side, it's definitely doable. We can add an assignment field in the task creation form and display the assigned user on each task.

Chloe: Agreed, and we could also have a color-coded indicator or avatar next to assigned tasks to make it visually clear who each task belongs to.

Raj: I see where the idea is coming from, but adding assignment capabilities will require significant backend adjustments. We'd need to implement user management for each task, which involves additional authentication layers.

Mia: Yes, we'd essentially need to build out user profiles and permissions, which aren't part of the current infrastructure. This would also mean more time for data management, as each task would be linked to multiple users.

Bob: Good points. Sarah, do we know how crucial this feature is for our customers? This will push back our current timeline if we add it in now.

Sarah: I understand the concerns. From the customer's feedback, they'd really value it, but they're also open to having it in a later release if that's more practical.

David: This would also mean a lot of additional testing on our end, especially around security and access rights for assigned tasks. We'd need to handle edge cases like task reassignment, deleting users with assigned tasks, etc.

Sarah: Understood. Based on this, it sounds like we may need to set it as a post-MVP feature. Let's stick to the current scope for now, but we'll keep this on our roadmap. Does that work for everyone?

Bob: Yes, that sounds reasonable. We can document the requirements for the assignment feature and plan it for a future release after the MVP.

Liam: Agreed. This way, we can keep our current timeline and avoid introducing complex changes mid-development.

Sarah: Perfect. Thanks, everyone, for the input. I'll update the customer on our plan, and we'll circle back to this feature once we have the MVP in place.

Date: 5 February 2024

Sarah: Hi, everyone. Today's focus is on performance. We've been getting feedback that the app struggles to maintain speed when there are a lot of concurrent users. Let's discuss the issues and potential solutions.

Raj: Right, we've noticed that when there are many active users, response times are slowing down significantly. This seems to be due to the current way we're handling data fetches and task updates.

Mia: I agree. It looks like the database queries for loading tasks are becoming a bottleneck. With more users, the load increases exponentially, and our current setup isn't handling it well.

Emily: That makes sense. On the frontend, we've been seeing more lag in task updates, which is noticeable to users.

Liam: I think we should consider implementing caching on frequently accessed data to reduce the load on the database.

Raj: Caching might work, but it's not a one-size-fits-all solution here. It could lead to stale data issues, especially with real-time updates like task assignments.

Bob: Liam has a good point, though. A caching layer could reduce the load on the database, even if we use it selectively.

Mia: (sighs) I'm not sold on caching as a sustainable solution. We're just avoiding the core issue, which is that our current database structure wasn't designed for this kind of load. Caching would only add complexity and risk outdated data without fully solving the problem.

David: Mia's point is valid. Caching would mean additional testing to ensure data consistency. But if we're open to exploring other solutions, what about moving frequently accessed data to a faster, in-memory database like Redis?

Raj: Redis could help with performance, especially if we use it for specific data points that require frequent access. But implementing it would require reworking parts of our architecture.

Bob: Sarah, based on priorities, do we have a clear timeline for addressing this?

Sarah: We need to ensure smoother performance before our next milestone. Let's aim to get a working solution within a month. If Redis or caching is viable, we can go that route—but only if it doesn't significantly extend our timeline.

Mia: Fine, if we go with Redis, let's ensure it's well-integrated to avoid further complications down the line.

Sarah: Agreed. Let's have Raj and Mia prototype the Redis approach and assess feasibility. We'll revisit in two weeks to check progress.