Logic Document

Project: Real-Time Collaborative To-Do Board by Sabeer Anwer Meeran

Here’s a breakdown of how I thought about and implemented two of the key features in this application: the Smart Assign button and the system for handling editing conflicts.

My Approach to "Smart Assign"

When I thought about the Smart Assign feature, I wanted to build something simple that just worked. The main idea was to automatically find the person on the team who had the least on their plate and give them the new task.

How I Built It:

When someone clicks the “SA” button, the server first gets a list of all the registered users to make sure everyone is considered. Then, it looks at all the tasks that are still in the “Todo” or “In Progress” columns I decided to ignore “Done” tasks since that work is already finished. After that, its a simple counting game. My code goes through the active tasks and counts how many belong to each person. The person with the lowest number is the one who gets the new task. Finally, the system saves the assignment and sends out a real-time update via WebSockets so everyone's board refreshes instantly to show the new owner of the task.

Example: Imagine sabeer1 already has 3 tasks they're working on, while sabeer2 only has 1. If I click “Smart Assign” on a new, unassigned task, my code sees that sabeer2 has more availability and gives them the task.

2. My Solution for Conflict Handling

I knew that in a real-time collaborative app, its possible for two people to try to edit the same task at the same time. To prevent one person from accidentally overwriting another persons work, I built a conflict handling system.

How I Built It:

My solution was to use the updatedAt timestamp that the database automatically puts on every task. I decided to treat it like a simple version number. When the frontend wants to save a change, it doesnt just send the new data and it also includes the timestamp of the task as it appeared *before* the user started editing. I call this the lastKnownTimestamp.

When the backend receives the update request, the first thing it does is compare this lastKnownTimestamp from the user with the actual updatedAt timestamp of the task in the database. If they match everything is fine and the save goes through.

But if they don't match, it means someone else saved a change in the meantime. When that happens, my backend immediately stops the update process. It rejects the save

and sends back a special 409 Conflict error. This tells the user's app that there was a problem and prevents the old data from overwriting the newer changes, protecting everyone's work.

Example: Let's say Sabeer1 and Sabeer2 both open a task at 10:00 AM. Sabeer1 quickly makes a change and saves it at 10:01 AM. The tasks version in the database is now 10:01 AM. When Sabeer2 tries to save their change at 10:02 AM, they are still sending the old 10:00 AM timestamp. My backend sees the mismatch rejects the save and prevents Sabeer1's work from being lost.