1. Smart Assign – Logic Explanation

The Smart Assign feature was created to simplify the process of assigning tasks to team members in a balanced and fair way. Instead of manually selecting users, the system automatically evaluates the load on each user and chooses the one with the fewest active assignments.

How it works:

- Whenever the Smart Assign button is clicked on a task, the system gathers all users and all tasks currently in the database.
- It loops through the tasks to count how many are assigned to each user.
- It then identifies the user with the smallest number of currently assigned tasks.
- That user is chosen to be assigned to the task.
- The assignment is recorded in the task, and an activity log entry is created to show who triggered the Smart Assign and which user was selected.

This ensures that workload is distributed fairly and that no user is overloaded while others remain unassigned.

Why it's useful:

- Prevents manual bias or oversight
- Encourages even team participation
- Reduces project management overhead
- Can be scaled further to account for user skill level, task priority, or custom filters

Potential Future Enhancements:

- Weight-based load (e.g., tasks with higher priority count more)
- Availability status of users

- Deadlines and task duration estimates
- AI-based suggestions based on historical performance or task types

---

2. Conflict Handling – Logic Explanation

Conflict handling was introduced to deal with scenarios where two or more users try to edit the same task simultaneously. Without conflict resolution, changes can be unintentionally overwritten, leading to data loss or confusion.

How it works:

- When a task is opened in the modal, the frontend displays its current state (title, description, etc.).
- If another user updates the same task and saves it, a real-time sync pushes the update to all connected clients.
- If the first user now attempts to save their stale copy (which doesn't include the latest changes), the server compares the incoming update with the latest stored version.
- If there's a mismatch (e.g., the description or title was changed in between), the backend or frontend flags a conflict.
- Instead of blindly overwriting, the user is shown a dialog with two versions:
    - Their local (unsaved) version
    - The latest saved version from the database
- The user is then asked to choose one:
    - **Merge manually** (by copying/editing between both)
    - **Overwrite** (if they are sure their version is correct)
    - **Discard their changes** and keep the other version

Why it's useful:

- Protects against accidental overwrites
- Maintains collaboration integrity
- Gives full control to the user on how to resolve the issue

Example Scenario:

- User A opens Task #12 and starts editing the title to "Fix Login Bug".
- Meanwhile, User B edits the same task's description and saves it.
- Now User A tries to save their title change. Since the backend version has changed (description was updated), a conflict is detected.
- User A is shown both versions and can decide how to proceed.

Potential Future Enhancements:

- Auto-merge logic (e.g., if title and description were changed independently, allow both)
- Real-time lock system (warn users when another user is actively editing)
- Display user avatars/icons showing who is currently viewing or editing a task

---

Final Note

While the core logic for Smart Assign and Conflict Handling has been implemented successfully, there is definitely room for further refinement. Due to ongoing academic exams, I had only around three focused days to complete this project. Within that time, I prioritized delivering a fully functional, end-to-end working system that covered all core requirements of the assignment.

That said, some aspects—particularly Conflict Handling—have a very minimal UI and straightforward logic at this stage. In a more extended timeline, I would have explored more robust features like real-time collaborative editing, automatic field-level merging, or even task edit-lock indicators to enhance the user experience.