**View Current Task Status**  
  
In a real-world application, the process for employees to update their "current status of achievement" (like "Active Conv. to Prospects," "Active Conv. to Won," and "Active Sales") and for those updates to reflect on the admin dashboard would typically involve the following steps:

1. **Employee Dashboard Interface:**
   * Each employee would have a dedicated dashboard displaying their assigned tasks.
   * For each task, there would be editable fields (e.g., input boxes or forms) where the employee could input their current achievements for metrics like "Active Conv. to Prospects," "Active Conv. to Won," and "Active Sales."
   * A "Save" or "Update Progress" button would be associated with these fields on the employee's dashboard.
2. **API Submission (from Employee Dashboard):**
   * When an employee enters or updates a value and clicks "Save" on their dashboard, a client-side script (JavaScript) on their page would collect these updated achievement values.
   * This data would then be sent via an API call (e.g., a POST or PUT request to a /employee/tasks/update-progress endpoint) to the backend server.
   * The API request would include the taskId, the employeeId, and the updated achievement metrics.
3. **Backend Processing and Database Update:**
   * The backend server would receive the API request from the employee.
   * It would validate the data (e.g., ensure the employee is authorized to update that task, data types are correct).
   * The server would then update the corresponding task record in the central database with the new "active" achievement values.
4. **Admin Dashboard Reflection:**
   * **Real-time Updates (Ideal):** For real-time reflection, the backend could use technologies like WebSockets or server-sent events to push immediate updates to all connected admin dashboards viewing that task.
   * **Polling/Refresh (Common):** More commonly, when an admin views the "View Current Task Status" page on their dashboard, the page would make an API call to fetch the *latest* task data from the backend. This data would then include the most recent "active" achievement values submitted by the employees. Alternatively, the admin dashboard might periodically refresh the data (e.g., every few minutes) or have a manual "Refresh" button.

**TECH. STACK:**

**To make this fully functional, you would need to implement:**

* **Employee-facing application/interface:** A separate web page or component where employees can log in and view/update *their* tasks.
* **Backend API for employee updates:** New API endpoints that an employee's dashboard can call to submit their progress updates.
* **Database integration:** A proper database to persistently store and retrieve the "active" achievement values, rather than just static JavaScript arrays.