TaskLedger – Complete Project Specification

This document describes TaskLedger in full detail. It intentionally avoids high■level summaries and instead explains the system logic, permissions, workflows, backend philosophy, and dashboard visuals in a clear and explicit manner.

1. Core Concept and Purpose

TaskLedger is a task, activity, and performance tracking system designed to enforce accountability within structured organizations. Unlike typical task management tools that focus only on task lists, TaskLedger treats every task as a measurable unit of work whose lifecycle, ownership, and transitions must be traceable. The system is designed around the idea that work should be auditable and analyzable.

The word "Ledger" is central to the philosophy. Just as a financial ledger records every transaction, TaskLedger records every task action: creation, assignment, status change, reassignment, and completion. No action exists without an event record, and no event exists without an actor and timestamp.

2. Problems Addressed by TaskLedger

Many task systems fail when teams grow because they lack visibility, accountability, and measurable performance indicators. TaskLedger directly addresses these gaps by maintaining persistent records of who did what, when it was done, and how task states evolved over time.

The system is explicitly designed to answer operational questions such as who assigned a task, whether deadlines were respected, how long tasks remain blocked, and how individual productivity trends evolve over time. These questions are impossible to answer reliably in systems that overwrite task states without logging historical context.

3. Role Model and Access Control

TaskLedger enforces a strict, fixed role model. Roles are not configurable, extensible, or user-defined. This design choice eliminates ambiguity and simplifies permission enforcement at the backend level. All permission checks are enforced server-side using authentication, permission classes, and queryset restrictions.

ADMIN users possess global visibility and authority across the system. They are responsible for user creation, role assignment, department creation, and high-level analytics. ADMINs are not bound by departmental constraints and can audit actions across the organization.

MANAGER users operate within a defined department. They can create tasks within that scope, assign tasks to employees belonging to the same department, and monitor performance and progress metrics for their team. Managers are operational controllers but do not possess administrative or security authority.

EMPLOYEE users are execution-focused. They cannot create or assign tasks. Their interaction is limited to viewing assigned tasks, updating task status as work progresses, and recording comments or work logs. Employees have no visibility into analytics beyond their own personal performance metrics.

4. Task Lifecycle and Events

Tasks in TaskLedger follow a controlled lifecycle. States such as TODO, IN_PROGRESS, BLOCKED, COMPLETED, and CANCELLED are not merely labels; transitions between them are governed by rules. Certain transitions are disallowed to prevent inconsistent data, such as reverting completed tasks without explicit events.

Every state transition generates a TaskEvent. TaskEvents store the previous state, the new state, the user who initiated the change, and the exact time of the action. This ensures that task history is never lost and can be reconstructed for audits or analytics.

5. Backend Architecture Overview

TaskLedger is built using Django and Django REST Framework with a backend-first mindset. The backend is treated as a data service, not a template-rendering engine. All interactions occur through JSON-based APIs secured using JWT authentication with asymmetric signing (RS256).

Business rules are enforced primarily through serializers and permission classes. Views remain thin and focused on request flow and orchestration. Models represent persistent data and relationships, while serializers define the API contract and validate incoming data.

6. Dashboard Design Philosophy

The dashboard design in TaskLedger is role-specific. Each role sees only the data necessary to perform its responsibilities effectively. Visual clutter is intentionally avoided, and information density increases with privilege level.

The UI follows a dark, enterprise-oriented theme with restrained color usage. Accent colors are applied only to indicate status, alerts, or critical metrics. Layouts rely on consistent spacing, grid alignment, and predictable navigation structures.

7. ADMIN Dashboard Visual Layout

The ADMIN dashboard provides a system-wide overview. KPI cards at the top display total users, active tasks, department count, and overdue tasks. Below these, analytical charts visualize task distribution by status and task volume over time. This enables high-level health monitoring of the organization.

The lower section of the dashboard presents recent task events and audit highlights. These tables allow ADMIN users to identify unusual patterns, role changes, or operational risks without navigating multiple screens.

8. MANAGER Dashboard Visual Layout

The MANAGER dashboard is execution-focused. It emphasizes task flow within the department and provides visibility into team workload and bottlenecks. Managers can quickly identify blocked or overdue tasks and take corrective action.

Charts and tables display task completion rates per employee, average completion time, and task backlog. These analytics support operational planning rather than administrative oversight.

9. EMPLOYEE Dashboard Visual Layout

The EMPLOYEE dashboard prioritizes clarity and focus. Employees see only their assigned tasks and personal performance indicators. The main task list is sorted by priority and deadlines to help employees manage workload efficiently.

Personal analytics provide feedback on task completion trends without exposing peer or team-level data. This reinforces accountability without creating unnecessary competition or distractions.