Real-Time Task Manager - Project Requirements & Workflow

Project Overview

A **Real-Time Task Manager** that allows users to create, assign, and track tasks in real-time using WebSockets. The system includes authentication, background tasks, session-based user management, and caching for optimized performance.

Key Learning Areas

- Sessions & Cookies
- Storages
- Background Tasks (Celery, Celery Beat)
- Caching (Redis)
- WebSockets (Django Channels)
- Middleware
- Authentication (Django JWT)
- ORM (Complex Queries)
- Database Design
- Minimal UI (Focus on Backend)

Project Requirements

User Roles

- Admin: Manages users, tasks, and system settings.
- Manager: Assigns tasks and monitors progress.
- Employee: Completes assigned tasks and updates progress.

Authentication & User Management

- User registration & login using Django authentication.
- Middleware to restrict unauthorized access.
- Session-based authentication with cookies.

Task Management

- Create, update, delete, and assign tasks.
- Set task priority (Low, Medium, High, Critical).
- Attach files to tasks using Django storages.

Real-Time Updates (WebSockets)

- Notify users when tasks are updated or assigned.
- Display task progress updates in real-time.

Background & Periodic Tasks

- Celery worker to send email notifications.
- Celery Beat for scheduled daily task reminders.

Caching & Optimization

- Redis caching for frequently accessed task lists.
- Optimized ORM queries.

Analytics & Reports

- Generate reports on overdue tasks, completed tasks, and productivity.

Minimal UI Focus

- Basic Django templates will be used for demonstration purposes.

Workflow of the Project

1. Users register and log in.
2. Managers create tasks and assign them.
3. Employees update progress (Real-time via WebSockets).
4. Celery sends task reminders.
5. Redis caching improves performance.

6. ORM queries generate analytics.

Technologies Used

- Django (Backend Framework)
- Redis (Caching & WebSockets)
- Celery & Celery Beat (Background & Periodic Tasks)
- PostgreSQL/MySQL (Database)
- Django Channels (WebSockets for Real-Time Updates)
- AWS S3/Local Storage (File Storage)
- HTML, CSS, JavaScript (Minimal UI)

Expected Learning Outcomes

By completing this project, one will gain hands-on experience in:

- Implementing **JWT authentication**.
- Using **WebSockets** for real-time updates.
- Managing **Celery tasks** for automation.
- Optimizing **performance with caching**.
- Writing **complex ORM queries**.
- Designing **efficient database schemas**.
- Implementing **middleware for security & logging**.

This project ensures an in-depth learning experience focused on Django backend development.