Project Report: Task Manager Application 📝

Based on the provided diagrams, this is a **task management application** built on the Telegram platform. It streamlines project and task workflows, from assignment to completion and approval.

Core Functionality & Workflows 🔄

The system supports two main workflows: **Project Setup** and **Task Tracking & Approval**.

Project Setup

A manager can **create a new project**, **assign employees** to it, and **add specific tasks**. This establishes the initial structure for work.

Task Tracking & Approval

Employees can **track the completion** of their assigned tasks. Once a task is marked as complete, it's sent to the **assigned manager for approval**.

- If the manager **approves** the task, its status is **updated**, and the employee is **notified** of the approval.
- If the manager **denies** the approval, the employee is also **notified**, and the task may be reverted to a non-completed status for further work.

System Architecture & Components 🛠️

The application follows a modular architecture, with a clear separation of concerns.

01

User Interface

The application uses the **Telegram API** for all user interaction, functioning as a bot.

02

Main Application Logic (app.js)

This file serves as the core of the application, handling user requests received from the Telegram API. It routes these requests to different modules based on the user's input (commands or scenes).

03

Modules

The application's logic is organized into several distinct modules:

- **Commands Module**: Handles specific, one-time commands from users (e.g., /start, /create_project).
- **Scenes Module**: Manages multi-step user interactions, like creating a new project or reporting task completion.
- **Models Module**: Acts as the interface between the application logic and the database. It contains functions for querying and updating the database.
- **Reminders Module**: A separate component (scheduler.js) that runs independently to **fetch due tasks** from the database and **send automated reminders** to users via the Telegram API.
- **Utils Module**: Contains helper functions used across various modules to avoid code duplication.

04

Database

The application uses a **MySQL Database** to store all project, task, user, and reminder information.

Data Model (ER Diagram) 🗄️

The database is structured with several interconnected tables to support the application's functionality.

users	Stores user information, including id, telegram_id, username, first_name, last_name, and role. The role attribute is critical for differentiating between employees and managers.
projects	Stores details about each project, such as id, name, description, manager_id, start_date, and end_date. The manager_id is a **foreign key** linking back to the users table.
tasks	Stores individual task details, including id, name, description, employee_id, approved_by, due_date, status, and priority. The employee_id, approved_by, and project_id fields are **foreign keys** linking to the users and projects tables.
project_employees	A **junction table** that manages the **many-to-many relationship** between projects and employees. It links an employee_id to a project_id.
reminders	Stores information about scheduled reminders, including task_id, reminder_type, and a boolean field is_sent to track the status of the notification.

Overall, the application is well-designed with a logical flow, a robust database schema, and a clear component-based architecture, making it a functional and scalable solution for task management within a team.