TaskSphere – Modular Task & Project Management Backend API

1. Description:

Project Management System.

The goal of this project is to help students apply all the concepts learned throughout the course — from Node.js fundamentals to databases and clean architecture — by developing a complete and structured backend application without relying on any frontend knowledge.
In TaskSphere, users can manage projects, organize tasks, and collaborate with team members through RESTful API endpoints. The system will allow performing full CRUD operations on projects, tasks, and teams, while maintaining clean separation of layers and database abstraction using both MongoDB (with Mongoose) and SQL (with Sequelize).
Throughout the project, students will implement:

- **Core Node.js features such as file operations, path handling, and building HTTP servers.**
- **REST APIs with Express.js using MVC and Clean Architecture.**
- **Middleware for logging, validation, and error handling.**
- **Data persistence with both MongoDB and SQL, highlighting their differences.**
- **Basic HTML templates for reports or debugging using Node's file system — no frontend required.**

The final result will be a professional-grade backend system with realistic endpoints and modular architecture, ready for deployment or extension (e.g., adding authentication or frontend integration later).

Core Features

- **Project CRUD (create, update, delete, list projects)**
- **Task management under each project (track, update, filter by status)**
- **Team member management (assign roles and responsibilities)**
- **Logging system using Node.js fs module**
- **Report generation and summary endpoints**
- **Middleware-based validation and centralized error handling**

Technical Highlights

- **Node.js & Express.js for server logic and routing**
- **MVC + Clean Architecture for scalability and maintainability**
- **MongoDB (Mongoose) for flexible data models**
- **SQL (Sequelize) for structured data relations**
- **Custom Middleware for logging, validation, and error handling**
- **File System Operations for saving logs and reports**

2. Tasks to Complete the Project:

1. Project Management

- **Create, update, delete, and list projects.**
- **Each project includes:**
  - **Title**
  - **Description**
  - **Deadline**
  - **Team members (names, roles)**
- **Projects stored in MongoDB (using Mongoose).**

2. Task Management

- **Each project can contain multiple tasks.**
- **Each task includes:**
  - **Title, description**
  - **Assigned team member**
  - **Status (todo, in-progress, done)**
  - **Due date**

3. Team Management

- **Manage team members (add, update, delete, list).**
- **Assign members to projects or tasks.**
- **Team data stored in MongoDB for flexibility.**

4. Reports & Data Views

- **Generate analytics (e.g., number of tasks completed per project).**
- **Retrieve combined project-task data using both MongoDB and SQL layers.**
- **Export basic reports to .json or .txt files using Node's fs module.**

5. System Logging

- **Log every incoming request (method, endpoint, timestamp).**
- **Log system errors to a file (using fs).**
- **Optional: Display system stats (e.g., uptime, total requests) on an HTML status page.**

3- Folder structure example:

```
TaskSphere/
│
├── src/
│   ├── app.js                  # Express app initialization
│   ├── server.js               # Server startup (HTTP)
│   ├── config/                 # DB and environment configurations
│   ├── controllers/            # Handle incoming requests
│   ├── services/               # Business logic layer
│   ├── repositories/           # Data access layer (Mongo + SQL)
│   ├── models/                 # Mongoose and Sequelize models
│   ├── routes/                 # Express route definitions
│   ├── middlewares/            # Logging, validation, error handling
│   ├── utils/                  # File utilities, formatters, etc.
│   └── views/                  # HTML templates (optional)
│
├── logs/                       # Request and error logs
├── .env                        # Environment variables
├── package.json
└── README.md
```

---

4 - Project Deliverables – TaskSphere Backend API

1. Demo

A **live demonstration** of the backend system showing all main functionalities through **Postman** or **cURL** requests.
The demo should highlight:

- Project, Task, and Team CRUD operations.
- Request validation and error handling.
- Logging in the console and log file.

- MongoDB and SQL database integrations.
- Report generation (output file or HTML page).

💡 *No frontend is required — all testing and demo should be done via API calls.*

---

2. 🗂️ Project Source Code

A clean, modular, and well-structured Node.js project folder following **MVC + Clean Architecture**.
Must include:

- Controllers, Services, Repositories, Models, and Middleware directories.
- Configurations for MongoDB and SQL databases.
- Proper use of .env for environment variables.
- Log directory with generated log files.

---

3. 📘 Documentation

A professional README.md file containing:

- Project overview and purpose.
- Setup and installation steps.
- Database setup and configuration.
- List of all available API endpoints (with method, path, and brief description).
- Example API requests and responses (can include Postman screenshots).
- Description of folder structure and architecture layers.

---

4. 📄 Postman Collection

An exported **Postman collection** containing all API endpoints for testing:

- CRUD routes for Projects, Tasks, and Teams.
- Report endpoints.

- Example requests with sample data.

---

5. 📊 Reports Folder

Generated report files (using Node's fs module) saved in /reports/, including:

- Task summary report (summary.txt or .json).
- Optional HTML report (summary.html) for readability.

---

6. 🧰 Logs Folder

Contains system-generated logs (from the logger middleware):

- Request logs (request.log) showing all incoming HTTP requests.
- Error logs (error.log) showing system or validation errors.

Thank You
Edges For Training Team