Lead BE Engineer - PHP/Laravel - practical task

Hey there! 😊

Thanks so much for taking the time to work on this. Below, you'll find a simple task that we'd love for you to tackle. These are meant to be simple and fun, so don't stress—there's plenty of room for creativity, and nothing is set in stone.

Please don't spend more than 2 (or max 3) hours on these. Once you're done, just pop the code into a public GitHub repo and share the link with us. We're excited to see what you come up with!

Happy coding! 🚀

Context

You need to develop a mini-application for task management (TODO-list), where users can create, edit, delete, and filter tasks. Each task can be part of a project, which can also have its own participants.

Requirements

1. Architecture and Backend using Laravel

Create a Laravel project with a REST API for managing tasks and projects.

Implement user authentication using Laravel Sanctum or Passport (your choice).

Each user can have multiple projects, and each project can include multiple users.

Each task should have:

- Title
- Description
- Due date

- Task status (e.g., "new", "in progress", "completed")

Each project should have:

- Title
- Description
- List of participants (linked to users)

2. Filtering and Search

Add the ability to filter tasks by status and due dates.

Implement search by task title and description.

3. Database Handling

Use MySQL/PostgreSQL (your choice) for data storage.

Design proper relationships between tables (One-to-Many, Many-to-Many, etc.) and indexes (explain usage).

4. Frontend Interaction (Optional)

Provide a minimal frontend using Vue.js or React.js to interact with the API. Or just Postman collection or something like that to call those APIs

Ability to call following features:

- Adding a new task
- Editing an existing task
- Deleting a task
- Filtering tasks by status and due dates

5. Email Notifications for Overdue Tasks

After the task due date is reached, if the task is not marked as completed, send an email to the task's creator with a reminder that the deadline has passed. The email should be simple, containing the task title and a notification that the deadline has passed.

Use any available email service (you can use the sandbox, e.g. Postmark).

6. Reports

Implement a simple reporting system that generates summary information for each project:

- Number of tasks per project
- Percentage of completed tasks per project

7. Docker Deployment

The project should be containerized and deployed using Docker. Ensure the entire application (including database and services) runs within Docker containers.

8. Documentation

Write brief documentation explaining how to set up and run the project. Describe the architectural decisions you made and justify them.

Evaluation Criteria

- **Architecture:** Quality and justification of the chosen solutions.
- **Code Quality:** Code clarity, adherence to SOLID principles, proficiency with Laravel and PHP.
- **API Interaction:** Ease of interacting with the API.
- **Database Design:** Correctness and thoughtfulness of the data structure.
- **Email Notifications:** Proper setup and sending of email notifications.
- **Docker:** Proper configuration and deployment using Docker.
- **Documentation:** Clear explanations of the setup process and the architectural decisions made.