Assignment: Mini Leave Management System

Disclaimer: we believe that a great developer (intern or full time) does not only think about what to be built but rather also feels what is great for a customer (at least up to some extent). If there is a product that a developer is developing, he/she should know why and for whom it's being done and what are the possible edge cases (other than those given to the developer).

So, your evaluation will be based on the following parameters:

- How you are able to define and elaborate the problem statement in detail
- How you are able to think of more edge cases
- How you are able to logically think about the product and the solution
- How you are able to think end to end for a module and not just one feature
- Are you able to imagine and visualize as to how the product you are developing will look like to the user
- Are you able to think what flow / experience will be broken

Therefore, we would not only like to see / understand your technical skills in software development but also how you take up a project and obsess about delivering the best case possible from your side.

We would love to see and hear the logical product solutions (along with documentation) from you before you present your technical solutions.

Note: we don't care if your English is not great in explanation. We would love to hear your solutions in the language you are more comfortable in (English, Hindi, Tamil, Marathi, Telugu, Malayalam, Gujarati, Odia, Bengali, etc.).

Now, let's go to the problem statement:

Scenario:

You're building the MVP of a Leave Management System for a startup with 50 employees. The HR team should be able to:

1) Add employees with details (Name, Email, Department, Joining Date)
2) Apply, approve, and reject leave requests
3) Track leave balance for each employee.

Part 1 — Problem Solving & Core Development

Build APIs (or a simple console app) for:

- Adding a new employee
- Applying for leave
- Approving/Rejecting leave
- Fetching leave balance for an employee.

Edge cases to handle:

- Applying for leave before joining date.
- Applying for more days than available balance.
- Overlapping leave requests.
- Employee not found.
- Invalid dates (e.g., end date before start date).
- Mention any other edge-cases that you can think of.

Part 2 — High Level System Design

- Architecture diagram (Frontend, Backend, Database).
- How APIs and DB interact.
- How scaling will be handled if the company grows from 50 → 500 employees.

Bonus

- Deploy on free hosting (Heroku, Render, Vercel).
- Provide a README with setup steps, assumptions, and potential improvements.

Assessment Response Structure:

Candidates should submit their work in the following format:

- A single ZIP file containing source code, diagrams, and README.
- README should include: setup steps, assumptions, edge cases handled, and potential improvements.
- API endpoints with sample input/output and screenshots (if applicable).
- HLD diagram (class diagrams or pseudocode).
- If deployed, include the live URL and credentials (if any).