ProU Technology

Backend - API Development Assignment

Internship Coding Challenge – Track 2: Backend (API + Database)

1. Scenario

You are part of a backend team tasked with building the **Task Management API** for an internal company dashboard.
Frontend developers will later use your API to power their application (like the one from Track 1).
Your job is to **design and implement RESTful APIs** for managing **Employees** and **Tasks.**

The system should allow:

- Creating and listing employees
- Creating, updating, and retrieving tasks
- Linking tasks to employees
- Filtering tasks by status or employee

You can use **any backend language/framework** (Node.js/Express, Python/FastAPI, Java/Spring Boot, etc.).

2. Expected Outcome

A **fully functional REST API** that performs CRUD operations on:

- Employees
- Tasks

3. Deliverables

Please submit:

1. A **GitHub repository** with:
    - API Source code
    - Database schema or migration script
    - Sample data (SQL/JSON)
2. A **README.md** that includes:
    - Setup and run instructions
    - API endpoints list (with request & response examples)
    - Tech stack used
    - Any assumptions made

4. Instructions

- Use proper **REST API standards** (e.g., GET /tasks, POST /employees).
- Handle **validation** and **error responses** correctly.
- Use clear **models/entities** for Employee and Task
- Include at least one level of **relationship** (e.g., Employee ↔ Task)
- Implement **basic logging** and use appropriate **HTTP status codes**
- Write **modular, readable code** (split routes, models, config, etc.)

5. Focus Areas (Evaluation Criteria)

Area	What we look for
API Design	Clean, logical, RESTful endpoints
Code Quality	Readable, modular, error handling, consistent naming
Database Design	Proper schema, relationships, normalization
Data Validation	Input validation and handling of bad requests
Documentation	Clear instructions and endpoint details
Testing	Basic endpoint testing (Postman collection or script)

6. Bonus Challenge (Optional)

Add **authentication** (simple JWT or API key-based) and allow:

- Only authenticated users to create/update tasks
- Anyone to view tasks (public endpoint)

This demonstrates your understanding of real-world API security concepts.

7. Estimated Time

4 to 6 hours (depending on framework familiarity).

8. Example Schema

Employees Table:

Field	Type	Description
Id	INT(PK)	Unique employee ID
Name	VARCHAR	Employee name
Role	VARCHAR	Job role
Email	VARCHAR	Contact email

9. Example Endpoints

- GET /employees
- POST /employees
- GET /tasks?status=In%20Progress
- POST /tasks
- PUT /tasks/:id
- DELETE /tasks/:id