Pre-Interview Coding Exercise (Part 1)

Task Management API

Objective:

Develop a simple **Task Management System** with a focus on **backend development.**

Requirements:

At Kent & Essex, we commonly use the **MERN stack (MongoDB, Express.js, React, Node.js)** for backend and frontend applications, along with **Docker** for containerization.

However, you are encouraged to choose the tech stack you are most comfortable with. Our goal is to assess your **problem-solving skills** and **coding best practices,** not just familiarity with a specific framework. Please **explain your choices in your README.md** so we can understand your thought process.

1. Project Setup

- **Dockerize** the application (provide a Dockerfile).
- Include a **README.md** with setup instructions
- Use **Git** for version control with clear commits

2. Backend Development

- Use **Node.js (ExpressJS/NestJS) or Python (FastAPI/Django)**.
- Implement **CRUD operations** for tasks.
- Use a **relational (PostgreSQL, MySQL)** or **NoSQL (MongoDB)** database.
- Use **asynchronous processing** (e.g., background worker with Celery/RabbitMQ or BullMQ).

Bonus (Optional)

- Implement **authentication** (JWT, OAuth2, or API Key-based).
- Write **unit and integration tests** (e.g., Document each API endpoint with request/response examples.

Completing the Challenge

We understand that not every candidate will have experience with all aspects of this challenge—and that's okay! **The goal is not just to see what you already know, but also how you approach learning new concepts and adapting to challenges.**

If you come across something unfamiliar, such as **Kubernetes or CI/CD**, we encourage you to:

- Complete the parts you're confident in
- Research and attempt the parts you're less familiar with
- Document your thought process and learning journey

Even if you're unable to fully implement a particular section, **we value the effort you put into understanding it**. During the interview, we'll discuss **what you learned, your approach to problem-solving, and how you adapt to new situations.**

Above all, **don't be discouraged**—we all have different skill sets, and what matters most is your willingness to grow and explore new technologies!

Evaluation Criteria:

The following table outlines the key areas we assess during the coding challenge. Each criterion is designed to evaluate the candidate's technical skills, problem-solving approach, and adherence to best practices.

Category	Weight	Criteria	What We're Looking For
Code Quality	25	Clean, readable, well-structured.	• Follows best practices (e.g., modular, DRY). • Uses appropriate error handling and logging.
Functionality	25	API correctly implements CRUD operations.	• All endpoints function as expected. • Proper use of HTTP methods (GET, POST, PUT, DELETE).
Database Design	15	Well-structured schema and relationships.	• Uses the right database for the problem (SQL vs. NoSQL). • Sensible indexing for performance.
Version Control (Git)	10	Meaningful commit history.	• Uses clear commit messages. • Logical commits instead of dumping all changes at once.
Dockerization	15	Working Dockerfile provided.	• Application runs in a containerized environment.
Documentation	10	Clear setup instructions in README.	• README explains API usage and how to run the project.
Bonus Points	10	Authentication (JWT/API Keys), basic unit tests.	• Extra effort beyond minimum requirements.

Submission:

Submit your project by either zipping the files and emailing them or sharing a repository link for cloning. If you have any questions or need clarifications, feel free to reach out to us at coding-challenge@kemutual.com