Design Task – Shipment Management

Guidelines:

- The candidate should capture the results of the task in a feasible format and share with the Xgrid team member. Please note that the spirit of this test is to give you interesting technical challenges, and not consume your valuable time with fancy documentation. Screen captures are perfectly acceptable.
- Be prepared to explain your approach and outcome, if asked by a Xgrid team member
- Do not hesitate to ask questions if clarification is needed.
- Timelines: You have 7 days from the date of issue of test to complete it. You are welcome to submit your results early if you have completed the test. We understand that you will be completing these tasks while managing other work and personal commitments. So feel free to work with your Xgrid contact, if additional time is required. Please keep in mind that the general assumption we make in your evaluation is that you asked for the task and worked on it during your free time.
- Good luck and have fun! We hope you will enjoy the tasks as they are a brief overview of what you could be doing if you join our team at Xgrid. ☺

Tasks:

Candidate is required to provide a link to a private GitHub project (shared with 'xgrid-all' user). The project can be implemented in any language and framework that you choose; we of course do have a strong preference for Typescript, NodeJS/NestJS, Angular OR React. You can use any type of database with this task, you may be asked for the reason you choose the specific database type for this task.

The project is expected to be in a functioning form and relatively bug free. It should be independently deployable, with a dockerized implementation having an associated Dockerfile**.**

Candidate is required to implement a "Shipment Management" web application.

Minimal Requirements for Application:

- A backend that has few REST endpoints to modify a persistent storage.
- The database will have two modules: Users and Shipments.
- The Users will be of two types: Admins and Workers. Admins can add/remove workers and shipments, also they can assign shipments to workers. Workers can update shipments that they are assigned to. One shipment can be assigned to multiple workers.
- Create REST endpoints and implement functionality for all the actions mentioned above along with some basic endpoints like login/signup. You have to implement an RBAC functionality to ensure that users can only perform actions that they are allowed to according to the explained rules above.
- Each shipment will have a unique ID and a status field. (You can add more fields in the schema if needed).
- Implement an endpoint which a worker could call to update the status of a shipment. This endpoint should call an "**updateShipment**" function which would have the following rules and restrictions:
    - The function would take two arguments: ID and status.
    - **Whenever the function is called it will generate a random number between 1 and 60. The function will then wait for a number of seconds equal to the random number. After the timeout, the function will call the db to update the status of the shipment.**
    - **Also make sure that the executions of updateShipment with the same id never run concurrently (execution always in order and consecutive).**
    - Ensure that whenever the endpoint is hit the updateShipment function is called at least once with the corresponding ID and status passed to it.
    - Make sure that you keep in mind the scalability of the application while implementing this.
- A frontend that has enough of a UI to perform all the mentioned actions for admins and workers. A good design for the frontend UI will carry extra marks.

- Both the backend and frontend applications should be well structured and have good code quality.
- All the design decisions/explanatory parts of the solution should be part of a "README.md" file on the GitHub project. **This should be a private repository that is shared only with 'xgrid-all' user as a contributor.**

Other Requirements (Design Decisions)

***NOTE: Please incorporate these points if you think they are essential for the application. Specify why something is essential or why something is left out. If you cannot make that part of the application, please comment on each one of these issues and how you will tackle them.**

- Test driven development to validate your code. (Optional, works as a plus point)
- Travis CI / CircleCI integration. (Optional, works as a plus point)
- API documentation (Optional, works as a plus point)

Note that this design task is intentionally left open ended instead of dictating specifics. We would like to see what kind of design decisions and conventions you end up adhering to. One more thing we would like you to explain is that whether you think there are any problems/vulnerabilities/limitations in the submitted solution? If yes, please explain why.

Requirements:

Take all the shortcuts necessary to complete the task in time. The main aim of this exercise is to gauge the applicant's ability to use all the required tools to develop a functioning project. Scalability of the system should not be of much concern for this task, although the candidate is expected to provide his ideas for system architecture and scale during his technical interview.

The goal is to create a working demo with the necessary build setup. Prioritize showing a working demo of the setup over completing all the needed features. Create README/installation instruction for the tools being used.