PRODUCT CHALLENGE

Leverage delivers end-to-end supply chain management powered by artificial intelligence, helping answer the fundamental question “Where’s my stuff?” We’re tackling a trillion-dollar industry with a world-class team and are looking for others to join our team to help us scale.

How we review

We’re looking to understand how you approach building a software product meant to be used by many users and touch many systems (e.g. integrations).

The aspects of your approach we will assess include:

- **Architecture:** how clean is the separation between the front-end and the back-end?
- **Clarity:** clearly and concisely explain the problem and solution? Are technical trade offs explained?
- **Correctness:** If there is anything skipped, is there an explanation for why it is missing?
- **Code quality:** is the approach structured in a simple, easy to understand, and maintainable way? Is there any code smells or other red flags? Does the use of object-oriented code follow principles such as the single responsibility principle? Is the implementation of coding styles consistent with the language’s guidelines? Will it be consistent throughout the codebase?
- **Security:** are there any obvious vulnerabilities?
- **Testing:** how thorough are the proposed automated tests? Will they be difficult to change if the requirements of the application were to change? Are unit and some integration tests considered or included?
    - NOTE: We’re not looking for full coverage (given time constraint) but trying to get a feel for your testing skills.
- **UX:** Will the web interface be understandable and pleasing to use? Will the API be intuitive?
- **Technical choices**: Do choices of libraries, databases, architecture etc. seem appropriate for the chosen application?

Bonus points (these items are optional):

- **Scalability:** Will technical choices scale well? If not, is there a discussion of those choices?
- **Production-readiness:** Does the code include monitoring? Logging? Proper error handling? If so, how?

kylewhitaker51@gmail.com

Challenge description

You are creating a web application used for managing a **Contacts list for registered users**.

The application should handle the following workflows:

1. User registration with full name, email, and password
2. Display the current contact list for logged user
3. Add a contact to the list
4. Delete a contact item from the list

Contact details to be recorded are:

1. Full name
2. Email
3. Phone number

Your assignment consists of two parts:

1. Create project task breakdown to plan the development process from start to finish.
   Each task should include the following:
   a. Title
   b. Description
   c. Estimated time to complete (in hours)
   List of tasks can be delivered as Google Sheet or any other generally used format.

   NOTE: Please make sure to send the task breakdown list in advance of coding challenge.

2. Deliver the project code according to the description below.

Technology stack

The application is expected to implement the following:

1. Database: **PostgreSQL**
2. API: Restful **API**, based on **Node and ExpressJS** framework, that implements endpoints for:
    a. User registration
    b. Login
        i. User authorization should be by using JWT token in Authorization header for all secure endpoints
    c. Secured endpoint to get list of contacts for a used that is logged in
    d. Secured endpoint to create a contact for a used that is logged in
    e. Secured endpoint to delete a contact for a used that is logged in

3. Frontend: SPA web application based on **React with Redux and Saga**
    a. There is no specific requirement from a UI layout perspective and you are free to use any library/framework to create a UI layout
    b. Be mindful of input validation for email and phone number

Coding should follow the general **ES6 standard** with mandatory use of **asynchronous (async/await) methods**.

Bonus criteria

Additional points in evaluation will be given for:

1. Use of **TypeScript**
2. Use of **NestJS** with ExpressJS
3. Use of **TypeORM**
4. Email verification based on the standard flow:
    a. Send a unique email verification URL to the user
    b. Verify the user when the link is clicked from an email

kylewhitaker51@gmail.com

Delivery and time to complete

The code should be delivered as a Git repository uploaded to your personal account (any service is accepted and you are not limited to GitHub)

You can choose the format in which you help convey this during a 1-hour call to talk through architecture considerations, stack choices, task priorities, testing, and what you consider “feature complete.”

On average, we find the average candidate needs around 8-hours to have a 1-hour meaningful technical conversation. While not required, feel free to come prepared with materials to help present your approach and support conveying your engineering decisions.