Full-stack Tech Test - candidate version

Key points to test:

1. Advanced typescript usage - generics, type inference, etc
2. Scalability and performance - efficient db interaction, concurrency and parallel processing
3. Problem solving skills
4. Code quality and organisation

Code Section: TypeScript Challenges

- First of Array

  Implement a generic `First<T>` that takes an Array `T` and returns its first element's type

  ```
  type arr1 = ['a', 'b', 'c']
  type arr2 = [3, 2, 1]

  type head1 = First<arr1> // expected to be 'a'
  type head2 = First<arr2> // expected to be 3
  ```

- Trim string

  Implement `Trim<T>` which takes an exact string type and returns a new string with the whitespace from both ends removed.

  ```
  type trimmed = Trim<'  Hello World  '>
  ```

```
// expected to be 'Hello World'
```

- Get required

  Implement the advanced util type `GetRequired<T>`, which remains all the required fields

  ```
  type I = GetRequired<{ foo: number, bar?: string }>
  // expected to be { foo: number }
  ```

Code Section: TypeScript Trading platform Project

Overview:

Implement a trading platform where sellers can post and update **Deals** and buyers can retrieve them. When a **Deal** is posted to the platform, a new deal alert should also be sent to all the buyers who are authorised to see the seller's **Deals**.

Requirements:

- Sellers and Buyers can connect to this platform via a REST API
- Buyers should only be able see the deals of the sellers they are authorised to
- Buyers would need to get updates from any authorised sellers about any deals changes via a webhook, for example price change or status (sold, available)

Data Structures:

```
/// Deal
{
    "name": "Name",
    "seller": "REF",
    "total_price": 100,
    "currency": "gbp",
    "discount": { // optional
        "type": "flat or percentage",
        "amount": 100
    },
```

```
    "status": "available or sold",
    "items": [
        {
            "id": "ref",
            "name": "Bundle",
            "price": 100
        }
    ]
}
```

Key considerations:

- Please use Typescript, and Fastify or Hono for your server
- Please use Prisma to interact with database and make migrations
- Please write tests that cover the main flows and ideally at least some edge cases using Jest
- Both sellers and buyers should be using JWT to authenticate with API, this should be handled by a middleware
- Buyers shouldn't be able to access routes to create deals, this should be handled by a middleware
- Implement webhook notifications using Bull and Redis
- Please add a README - this doesn't have to have lots of information but should include brief instructions on how to:
    - Run the API locally
    - Run the tests

Nice to have:

- Add a global error handler to avoid having custom logic in each request handler
- Webhook requests need to be signed
- Open API Spec

No-Code Section: System Design

How would you deploy the system above? It needs to meet the following requirements:

- Updates to Buyers need to be delivered at least once
- Latency between receiving an update and sending it out to Buyers should be as low as possible

In the future this platform would need to be able to accept bids from buyers and send them invoices upon sale end. How would you implement it?

What other challenges do you see with this system? How would you approach them?