# Frontend Developer Hiring Test

## About This Test

We have prepared this assignment to ensure our company is the best fit for you. Once you have submitted your solutions, our engineers will carefully review them and provide feedback as promptly as possible.

## General Rules

- There is no time limit for completion. However, you need to submit your answers before proceeding to the 2nd interview.
- You are free to reference materials available on the internet. If you incorporate code that is not your original work, please attribute the author and/or source using comments.
- Please create a public GitHub repository that includes all your answers and share its URL with us. The repository should include a `README.md` file with a brief explanation of your implementation and instructions on how to run the code.

## Process After Submission

- Discussion session (30~60 minutes): We will conduct a discussion with you regarding the implementation you have submitted.

## Privacy Policy

- Your information will only be used for recruitment purposes at BEES. You are prohibited from making the contents of this test public in any way.

- Should you have any questions, feel free to contact us at: hr@thebees.group.

# The Tests

## 🧠 Logic Test

Create a function `processWithDelay` that demonstrates understanding of modern async patterns.

```
async function processWithDelay(numbers: number[]): Promise<void> {
  // Your implementation here
}
```

### Core Functionality

- The function should process each number in the array with a 1-second delay between each number. The function should print each number to the console when it is processed.
- The function should return a Promise that resolves when all numbers have been processed.
- The function should handle empty arrays gracefully and return a resolved Promise immediately.
- The function should also handle invalid inputs (e.g., non-array, non-numeric values) by throwing a custom error.

### Example

**Input:**

```
processWithDelay([1, 2, 3, 4, 5])
```

**Output:**

```
// After 1 second
1
// After 2 seconds
2
// After 3 seconds
3
// etc...
```

## Bonus Points ✨

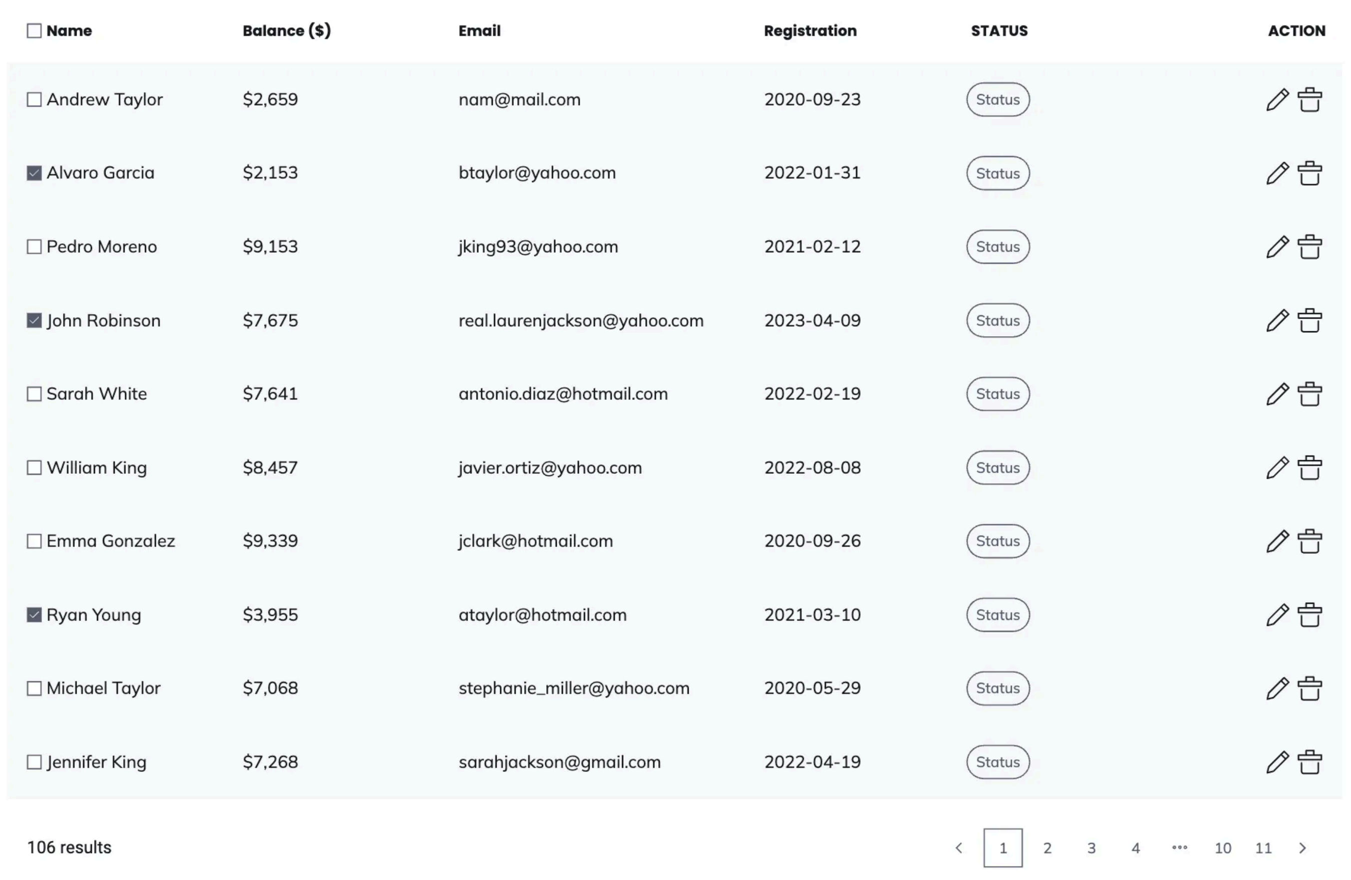
- Allow configurable delay time
- Support cancellation of the ongoing process
- Implement progress tracking

# 💻 App Development Test

You have been assigned to the team developing a users management page. Develop a users table that satisfies the requirements as mentioned in the tasks.

## Core Functionality

Referring to the image below, develop the core functionality of the users table. Specifications are written in order of priority.



| ☐ Name | Balance ($) | Email | Registration | STATUS | ACTION |
|---|---|---|---|---|---|
| ☐ Andrew Taylor | $2,659 | nam@mail.com | 2020-09-23 | Status | ✎ 🗑 |
| ☑ Alvaro Garcia | $2,153 | btaylor@yahoo.com | 2022-01-31 | Status | ✎ 🗑 |
| ☐ Pedro Moreno | $9,153 | jking93@yahoo.com | 2021-02-12 | Status | ✎ 🗑 |
| ☑ John Robinson | $7,675 | real.laurenjackson@yahoo.com | 2023-04-09 | Status | ✎ 🗑 |
| ☐ Sarah White | $7,641 | antonio.diaz@hotmail.com | 2022-02-19 | Status | ✎ 🗑 |
| ☐ William King | $8,457 | javier.ortiz@yahoo.com | 2022-08-08 | Status | ✎ 🗑 |
| ☐ Emma Gonzalez | $9,339 | jclark@hotmail.com | 2020-09-26 | Status | ✎ 🗑 |
| ☑ Ryan Young | $3,955 | ataylor@hotmail.com | 2021-03-10 | Status | ✎ 🗑 |
| ☐ Michael Taylor | $7,068 | stephanie_miller@yahoo.com | 2020-05-29 | Status | ✎ 🗑 |
| ☐ Jennifer King | $7,268 | sarahjackson@gmail.com | 2022-04-19 | Status | ✎ 🗑 |

106 results

< 1 2 3 4 ••• 10 11 >

1. Details of the users are shown in a multi-page table, with defaults 10 rows per page (user can change this).

2. When displaying in the table:

    1. Balance should have a thousand separator and be preceded by a "$" sign.

   2. Email should be a clickable link.
   3. Registration should be in “yyyy-MM-dd” format. When hovering with the mouse, it should show the detailed date and time, including hours & seconds.
3. The data schema of each row should be:

```
interface TUser {
  id: string
  name: string
  balance: number
  email: string
  registerAt: Date
  active: boolean
}
```

4. You are free to provide any sample data to render the table (at least 100 rows should be sufficient).

## Requirements

- You can use any frontend framework (Vue, React, Angular, Svelte, Solid, etc...) or none at all.
- TypeScript is mandatory.
- There is no specification for color or size of elements, but the design should be responsive.

## Bonus Points ✨

- Implement sorting and filtering functionality
- Implement a dark mode toggle
- Implement a virtualization mode (infinite scroll)
- Fetch data from a public API
  - Implement a loading spinner
  - Properly handle errors and display error messages to the user (when failed to fetch data)