3606. Coupon Code Validator

Solved

Easy | Topics | Companies | Hint

You are given three arrays of length `n` that describe the properties of `n` coupons: `code`, `businessLine`, and `isActive`. The `i^th` coupon has:

- `code[i]`: a **string** representing the coupon identifier.
- `businessLine[i]`: a **string** denoting the business category of the coupon.
- `isActive[i]`: a **boolean** indicating whether the coupon is currently active.

A coupon is considered **valid** if all of the following conditions hold:

1. `code[i]` is non-empty and consists only of alphanumeric characters (a-z, A-Z, 0-9) and underscores (_).
2. `businessLine[i]` is one of the following four categories: `"electronics"`, `"grocery"`, `"pharmacy"`, `"restaurant"`.
3. `isActive[i]` is **true**.

Return an array of the **codes** of all valid coupons, **sorted** first by their **businessLine** in the order: `"electronics"`, `"grocery"`, `"pharmacy"`, `"restaurant"`, and then by **code** in lexicographical (ascending) order within each category.

Example 1:

> **Input:** `code = ["SAVE20","","PHARMA5","SAVE@20"], businessLine = ["restaurant","grocery","pharmacy","restaurant"], isActive = [true,true,true,true]`
>
> **Output:** `["PHARMA5","SAVE20"]`
>
> **Explanation:**
>
> - First coupon is valid.
> - Second coupon has empty code (invalid).
> - Third coupon is valid.
> - Fourth coupon has special character `@` (invalid).

Example 2:

> **Input:** `code = ["GROCERY15","ELECTRONICS_50","DISCOUNT10"], businessLine = ["grocery","electronics","invalid"], isActive = [false,true,true]`
>
> **Output:** `["ELECTRONICS_50"]`
>
> **Explanation:**
>
> - First coupon is inactive (invalid).
> - Second coupon is valid.
> - Third coupon has invalid business line (invalid).

Constraints:

- `n == code.length == businessLine.length == isActive.length`
- `1 <= n <= 100`
- `0 <= code[i].length, businessLine[i].length <= 100`
- `code[i]` and `businessLine[i]` consist of printable ASCII characters.
- `isActive[i]` is either `true` or `false`.

Python:

```
class Solution:
    def validateCoupons(self, code, line, active):
        line_id = {
            "electronics": 1,
            "grocery": 2,
            "pharmacy": 3,
            "restaurant": 4,
        }

        def ok_chars(s):
            return all(ch.isalnum() or ch == "_" for ch in s)

        def is_valid(i):
            if not active[i]:
                return False
            if not code[i]:
                active[i] = False
                return False
            if line[i] not in line_id:
                active[i] = False
                return False
            if not ok_chars(code[i]):
                active[i] = False
                return False
            return True

        valid = [i for i in range(len(code)) if is_valid(i)]
        valid.sort(key=lambda i: (line_id[line[i]], code[i]))
        return [code[i] for i in valid]
```

JavaScript:

```
/**
 * @param {string[]} code
 * @param {string[]} businessLine
 * @param {boolean[]} isActive
 * @return {string[]}
 */
var validateCoupons = function(code, businessLine, isActive) {
    const exp = /^\w+$/;
    const businessLineMap = {'electronics': [], 'grocery': [], 'pharmacy': [], 'restaurant': []};
    const ans = [];

    for (let i = 0; i < code.length; i++) {
```

```
        if (exp.test(code[i]) &&
            businessLineMap[businessLine[i]] &&
            isActive[i]) businessLineMap[businessLine[i]].push(code[i]);
    }

    for (key in businessLineMap) {
        ans.push(...businessLineMap[key].sort());
    }

    return ans;
};
```

Java:

```
class Solution {
    public List<String> validateCoupons(String[] code, String[] line, boolean[] active) {
        int n = code.length;

        Map<String, Integer> lineId = new HashMap<>();
        lineId.put("electronics", 1);
        lineId.put("grocery", 2);
        lineId.put("pharmacy", 3);
        lineId.put("restaurant", 4);

        List<Integer> valid = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            if (!lineId.containsKey(line[i]) || code[i] == null || code[i].isEmpty()) {
                active[i] = false;
            }
            if (active[i]) {
                for (char c : code[i].toCharArray()) {
                    if (!isGoodChar(c)) {
                        active[i] = false;
                        break;
                    }
                }
            }
            if (active[i]) {
                valid.add(i);
            }
        }

        valid.sort((i, j) -> {
            int li = lineId.get(line[i]);
```

```
            int lj = lineId.get(line[j]);
            if (li != lj) return Integer.compare(li, lj);
            return code[i].compareTo(code[j]);
        });

        List<String> ans = new ArrayList<>(valid.size());
        for (int idx : valid) {
            ans.add(code[idx]);
        }
        return ans;
    }
    private boolean isGoodChar(char c) {
        return Character.isLetterOrDigit(c) || c == '_';
    }
}
```