

Boardgame App Code:

I've developed a board game app using React with Vite and IndexedDB, featuring multiple interactive pages such as the Mystery Box page for unlocking random rewards, the Side Quest page for optional challenges, and the Question page, which presents tasks or trivia essential to progressing in the board game.

Initial Version:



```
1  getAllMysteryBoxes().then(storedBoxes => {
2    if (!storedBoxes || storedBoxes.length === 0) {
3      const initialBoxes = generateMysteryBoxes();
4      saveAllMysteryBoxes(initialBoxes).then(() => {
5        setBoxes(initialBoxes);
6        setLoading(false);
7      });
8    } else {
9      setBoxes(storedBoxes);
10     setLoading(false);
11   }
12 });
13 }, []);
14
15 const openBox = async (box) => {
16   if (box.isOpened) return;
17   const revealedItem = getRandomItem();
18   const updatedBox = { ...box, revealedItem, isOpened: true };
19   await saveMysteryBox(updatedBox);
20   setBoxes(boxes.map(b => b.id === box.id ? updatedBox : b));
21 };
```

Feedback:

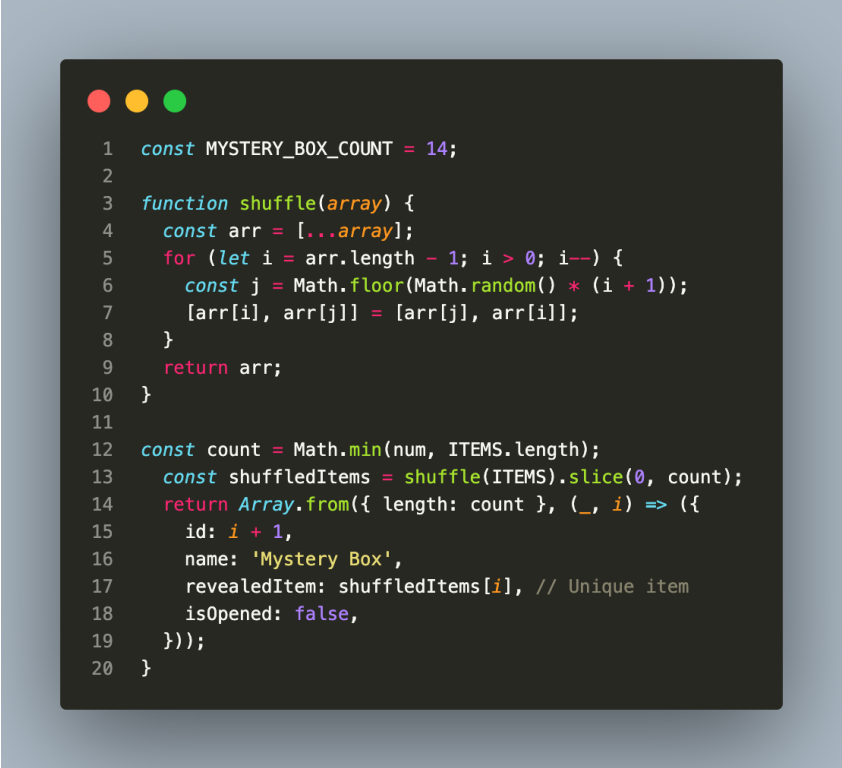
During testing with our peers, we noticed an issue with the website's code. Currently, it retrieves data from the database without properly randomising it. This results in the same data item being selected multiple times — for example, some items appear four times — while others aren't shown at all.

As feedback we received, we need to fix this. The system should be adjusted to fetch all data entries once, randomise the entire dataset, and then display each item only once. This ensures every data item is shown, and none are repeated unnecessarily.

What I plan to do:

I will structure the code to fetch all the data once and then randomise the positions on the front-end, instead of randomising the data and fetching the same item multiple times.

Iteration 1:



```
1  const MYSTERY_BOX_COUNT = 14;
2
3  function shuffle(array) {
4    const arr = [...array];
5    for (let i = arr.length - 1; i > 0; i--) {
6      const j = Math.floor(Math.random() * (i + 1));
7      [arr[i], arr[j]] = [arr[j], arr[i]];
8    }
9    return arr;
10 }
11
12 const count = Math.min(num, ITEMS.length);
13 const shuffledItems = shuffle(ITEMS).slice(0, count);
14 return Array.from({ length: count }, (_, i) => ({
15   id: i + 1,
16   name: 'Mystery Box',
17   revealedItem: shuffledItems[i], // Unique item
18   isOpened: false,
19 }));
20 }
```

What have I changed?

Previously, the code fetched and randomised data multiple times, which could result in the same item being retrieved more than once.

Now, you've changed it to **fetch all the data once**, store it, and then **randomise only the positions on the front-end**. This ensures all items are unique and only retrieved once, improving both performance and consistency.

Feedback:

This code is functioning well and displays the content as expected. After a reset, it properly re-randomises the data, ensuring a fresh and dynamic experience each time.

Reflection:

After coding both versions, I've realised that I now have the fluency to understand and solve problems more effectively. The changes I made to the code are implemented properly, and the way it displays and functions on the front-end shows that everything works as intended.

First and final version of the Code:

```

1  getAllMysteryBoxes().then(storedBoxes => {
2    if (!storedBoxes || storedBoxes.length === 0) {
3      const initialBoxes = generateMysteryBoxes();
4      saveAllMysteryBoxes(initialBoxes).then(() => {
5        setBoxes(initialBoxes);
6        setLoading(false);
7      });
8    } else {
9      setBoxes(storedBoxes);
10     setLoading(false);
11   }
12 });
13 }, []);
14
15 const openBox = async (box) => {
16   if (box.isOpened) return;
17   const revealedItem = getRandomItem();
18   const updatedBox = { ...box, revealedItem, isOpened: true };
19   await saveMysteryBox(updatedBox);
20   setBoxes(boxes.map(b => b.id === box.id ? updatedBox : b));
21 };

```

```

1  const MYSTERY_BOX_COUNT = 14;
2
3  function shuffle(array) {
4    const arr = [...array];
5    for (let i = arr.length - 1; i > 0; i--) {
6      const j = Math.floor(Math.random() * (i + 1));
7      [arr[i], arr[j]] = [arr[j], arr[i]];
8    }
9    return arr;
10 }
11
12 const count = Math.min(num, ITEMS.length);
13 const shuffledItems = shuffle(ITEMS).slice(0, count);
14 return Array.from({ length: count }, (_, i) => ({
15   id: i + 1,
16   name: 'Mystery Box',
17   revealedItem: shuffledItems[i], // Unique item
18   isOpened: false,
19 }));
20 }

```