

# Practice Challenge: Sushi Saga

 <https://github.com/learn-co-curriculum/react-hooks-practice-sushi-saga>  <https://github.com/learn-co-curriculum/react-hooks-practice-sushi-saga/issues/new>

Welcome to Sushi Saga, where your journey to sushi is only just beginning!

We've had a bit of trouble with our patented Sushi Saga conveyor belt system, so before you can eat, you're going to have to help us get it working.

**Here's what it should look like:**



**Tako Supreme - \$20**



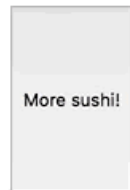
**Tsunndere Ebi - \$15**



**Oh Sake - \$10**



**Genki Toro - \$25**



**You have: \$100 remaining!**



**Doesn't that look delicious?!**

# Setup

All the sushi data about can be found in the `db.json` file. We'll be using `json-server` to create a RESTful API for our database.

Run `npm install` to install our dependencies.

Then, run `npm run server` to start up `json-server` on `http://localhost:3001`.

In another tab, run `npm start` to start up our React app at `http://localhost:3000`.

Note that there are no tests for this practice challenge, so you'll need to compare your app's functionality to the demo image above to ensure you've completed the deliverables successfully.

## Starter Code

Just as all good sushi needs a firm base of delicious rice, we've given you quite a bit of code to start off your frontend!

Inside this repo are all the components you'll need, as well as instructions as to where and how to render those components properly.

The component hierarchy should be as follows:

- `App` is parent to both `SushiContainer` and `Table`
- `SushiContainer` is parent to both `Sushi` and `MoreButton`

Be sure to read all of the notes in the all of the components before getting started! They will give you clues as to how and where to manage `state` and `props`.

## Deliverables

Inspectors will be coming by to check that our patented Sushi Saga conveyor belt is working properly! Oh no! They will be checking the following:

1. The sushi list is properly received from the server and displayed in our app.
2. Only 4 sushi are rendered at a time.

3. Clicking the "More Sushi!" button shows the next set of 4 sushi in the list. For this assignment, you don't have to be concerned about what happens when you reach the end of the sushi list.
4. Clicking a sushi on a plate will eat the sushi, causing it to be removed from its plate and an empty plate to appear on the table.
5. We need to make money! Whenever a sushi is eaten, customers should be automatically charged! Based on a budget decided by you, the developer, the amount of money remaining should go down by the cost of the sushi that was eaten. There is a spot to display this number in the `Table` component.
6. No free meals! Customers cannot eat any sushi that exceeds the amount of money remaining in their balance.

## Bonus

If and only if you have time, you may work on the following:

1. Sushi Wallet! Add a form for customers to add more money to their balance.
2. Full rotation! When the end of the line of sushi is reached, the conveyor belt should start from the beginning. Sushi that have already been eaten should remain eaten. It would be creepy if they reappeared!
3. Anything else!

**Note:** If at the end of the challenge you have achieved all the functionality required but the style looks off, this is okay!