RESTAURANT FAVES

Full Stack Angular & REST API

Task: Your friend is frustrated because they keep forgetting what they ordered last time they went to a restaurant and whether they liked it. With all the skills you've learned as a developer, let's make an app to solve their problem. Build an Angular front-end for the REST API we've already built to store a list of restaurant orders with ratings.

THE API:

Start up the existing API built with Spring Boot. ***NOTE: This API is already made. You only need to make an Angular app for it.*** If you don't already have a working version of the Restaurant Faves API, get the code here and follow the setup instructions in the readme.md. View the API at http://localhost:8080/orders.

The API stores order objects that look like this:

```
{
  id: 1,
  description: "Eggplant Parmesan",
  restaurant: "Antonio's",
  rating: 2,
  orderAgain: false
}
```

The API has the following endpoints:

- **GET /orders** - Responds with an array of all orders. It also includes two optional query string parameters for filtering:
  - restaurant (a string) - if specified, includes orders where the restaurant field contains the given text
  - orderAgain (true or false) - if specified, includes only orders that have the orderAgain field matching this value
- **GET /orders/{id}** - Responds with a single order object matching the given ID.
- **POST /orders** - Adds an order to the database. The order is specified as the JSON body of the request.
- **PUT /orders/{id}** - Updates an order in the database. The order is specified as the JSON body of the request.
- **DELETE /orders/{id}** - Removes the specified order from the database.

THE ANGULAR APP:

Setup:

Follow the setup instructions for Assessment 8. Name the Angular project **restaurant-faves-frontend**.

Build Specifications:

- Must include an Order Interface, which consists of the following properties:
  - **id** (number), *optional*
  - **description** (string)
  - **restaurant** (string)
  - **rating** (number)

    - **orderAgain** (boolean)
- Must include a restaurant-faves service that makes http requests (get, post and delete) based on the API's endpoints with the base URL of: http://localhost:8080/orders
- No CSS is required.

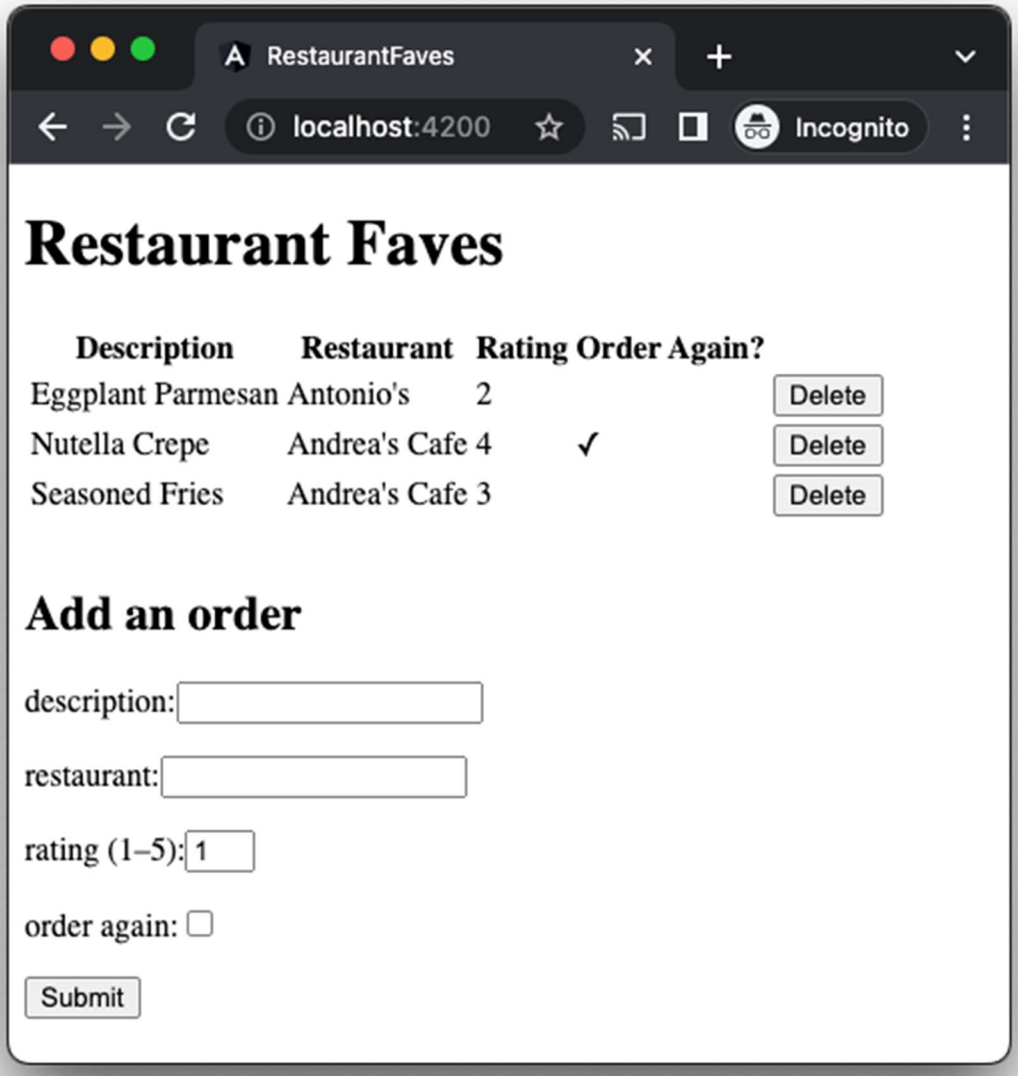
Components:

- **order-history component**: This displays an HTML table of results from the API and holds the add-order-form component.
    - Use *ngFor to display a separate table row for every one order.
    - Use *ngIf within each row to show a checkmark only when orderAgain is true. You can copy-paste this symbol for the checkmark: ✓
    - Include a button in each row to delete orders.
    - Include order-history in your app component so that it is displayed on the page.
- **add-order-form component**: Display a form that contains an input for restaurant, description, rating (1–5) and orderAgain. When the form is submitted, construct a new object based on the Order interface. The API will handle adding a new id. This component will need the following output property:
    - @Output() orderSave = new EventEmitter<Order>(); - notifies that the form was submitted and passes a new Order object based on the form input.

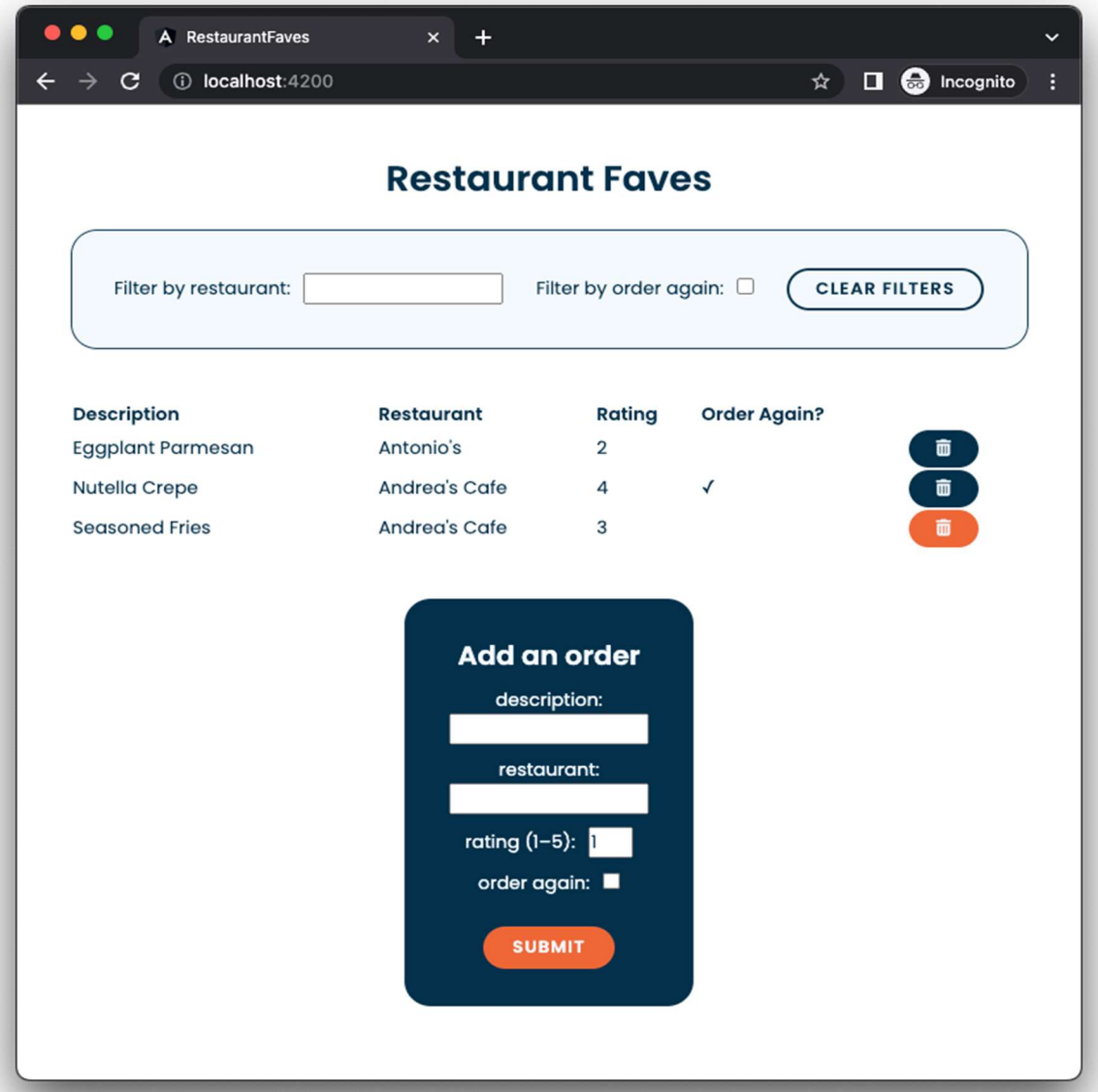
Extended Challenges:

- Add the ability to filter by restaurant and/or show only items with orderAgain set to true. Use the API's query parameters to accomplish this.
    - HINT: In the screenshot, the filters are applied as soon as the user types rather than submitting a filter form. To make this work, use the (input) event for textboxes and the (change) event for checkboxes.
- Add CSS. Try to emulate as much of the styling in the extended challenge sample screenshot as you can.

Sample Screenshots:
This screenshot shows is the basic required functionality:



This screenshot includes both the filtering and CSS extended challenges!



Style Guide:

- Font: Poppins at weights 400 (regular) for body text and 600 (semi-bold) for headings
- Colors:
  - #03324c (dark blue) for the text, borders and delete buttons
  - #f2f8fc (pale blue) for filter-form background color and add-order-form text
  - #ef6738 (orange) for hover states on the trash icon buttons and the add-order-form submit button.
- Trash icons: Font Awesome (be sure to link your kit in the head of the html file).