Charter/Spectrum Front-End Code Challenge

For this challenge we would like you to create a React application that pulls restaurant data from a simple API, displays that data in a table, and allows users to filter that data.

API Endpoint: https://code-challenge.spectrumtoolbox.com/api/restaurants
API Key Header: Authorization | Api-Key q3MNxtfep8Gt

Example Fetch:

```
fetch("https://code-challenge.spectrumtoolbox.com/api/restaurants", {
    headers: {
      Authorization: "Api-Key q3MNxtfep8Gt",
    },
});
```

User stories are as follows:

- A user should be able to see a table with the name, city, state, phone number, and genres for each restaurant.
- A user should see results sorted by name in alphabetical order starting with the beginning of the alphabet
- A user should be able to filter restaurants by state. If a state is selected that does not contain any restaurants, there should be something that indicates no restaurants were found for that state.
- A user should be able to filter by genre.
- State and Genre filters should default to "All" and take effect instantaneously (no additional clicks).
- A user should be able to enter text into a search field. When hitting the enter key or clicking on a search button, the table should search results. Search results should match either the name, city, or genre.
- A user should be able to clear the search by clearing the text value in the search input.
- A user should only see 10 results at a time and the table should be paginated.
- A user should be able to combine filters and search. The user should be able to turn filters on and off while a search value is present.

What we are looking for:

- No use of third-party libraries for the table/filter/search. Using Create-React-App or Next.js as a starter kit is okay.
- Well organized file structure
- Descriptive naming conventions
- DRY code that is readable and production ready
- Reusable components
- Sound logic for how the filters are architected
- Styling follows a convention/pattern and is well organized
- Full Git history with atomic commits

Stretch goals:

- Deployed application
- CI / CD
- Unit tests
- TypeScript
- Table row click shows additional information
- Add filter for attire
- Feel free to get creative!