🔒 **FullstackAcademy** / **jpfp-requirements**   Private

The requirements for JPFP.

☆ **0** stars   ⑂ **12** forks

☆ Star	⊙ Watch ▾

`<> `**Code**   ⓘ Issues   ⇅ Pull requests   ⊳ Actions   ▥ Projects   📖 Wiki   ⊘ Security

⑂ **main** ▾   • • •

| 🏔 **ThompsonHarris** formating   ⋯ | 6 days ago   🕓 **4** |

View code

README.md

JPFP

Requirements

The requirements below are broken into separate **tiers**, which model the way we **recommend you approach the project**. That is, we recommend you complete (or complete the majority of) the requirements in Tier 1 before moving on to Tier 2, and so on. Of course, if you get stuck on a particular feature, we recommend moving on and trying another feature - don't sacrifice the good for the perfect!

[See some wireframes here!](#)

Optional Boilerplates to start! 🌶

- [Spicy](#)
- [Medium](#)
- [Mild](#)

Tiers

Tier 1: All Campuses and Students (20/61)

▶ Details

Tier 2: Single Student and Single Campus (12/61)

▶ Details

Tier 3: Adding a Campus and Adding a Student (10/61)

▶ Details

Tier 4: Removing a Campus and Removing a Student (8/61)

▼ Details

Frontend

✏️ In the all-campuses view, include an `X` button next to each campus

- Clicking the `X` button should:

  ✏️ Make an AJAX request that causes that campus to be removed from database
  ✏️ Remove the campus from the list of campuses without needing to refresh the page

✏️ In the all-students view, include an `X` button next to each student

- Clicking the `X` button should:

  ✏️ Make an AJAX request that causes that student to be removed from database
  ✏️ Remove the student from the list of students without needing to refresh the page

Backend

✏️ Write a route to remove a campus (based on its id)
✏️ Write a route to remove a student (based on their id)

Congrats! You have completed your fourth vertical slice! Make sure to `commit -m "Feature: Remove Campus and Student"` before moving on (see RUBRIC.md - points are awarded/deducted for a proper git workflow)!

Tier 5: Updating a Campus and Updating a Student (11/61)

▼ Details

Frontend

☑ Write a component to display a form updating *at least* a campus's name and address

☑ Display this component as part of the single-campus view, alongside the single campus

- Submitting the form with valid data should:

  ☑ Make an AJAX request that causes that campus to be updated in the database
  ☑ Update the campus in the current view without needing to refresh the page

☑ In the single-campus view, display an `Unregister` button next to each of its students, which removes the student from the campus (in the database as well as this view); hint: the student is still in the database but is no longer associated with the campus

☑ Write a component to display a form updating *at least* a student's first and last names, and email

☑ Display this component as part of the single-student view, alongside the single student

- Submitting the form with valid data should:

  ☑ Make an AJAX request that causes that student to be updated in the database
  ☑ Update the student in the current view without needing to refresh the page

Backend

☑ Write a route to update an existing campus
☑ Write a route to update an existing student

Bonus Tier: Finishing Touches (15 EC)

▼ Details

Testing

- Write the following tests, each marked with a *** in the tests directory

Do these first ☐ React (AllCampuses): renders "No Campuses" if passed an empty array of campuses

☐ React (AllStudents): renders "No Students" if passed an empty array of students

☐ Redux (campuses): returns the initial state by default

☐ Redux (students): returns the initial state by default

☐ Express: GET /api/students responds with all students

☑ Sequelize (Campus): requires name and address

☐ Sequelize (Student): email must be a valid email

☐ Navigation: navbar to navigate to home, campuses, students

Finishing Touches

☑ If a user attempts to add a new student or campus without a required field, a helpful message should be displayed

☐ If a user attempts to access a page that doesn't exist (ex. `/potato`), a helpful "not found" message should be displayed

☐ If a user attempts to view a student/campus that doesn't exist, a helpful message should be displayed

☐ Whenever a component needs to wait for data to load from the server, a "loading" message should be displayed until the data is available

☐ Overall, the app is spectacularly styled and visually stunning

Ordering

☐ Create option for students to be ordered based on lastName on all-students view

☐ Create option for students to be ordered based on GPA on all-students view

☐ Create option for campuses to be ordered based on number of enrolled students on all-campuses view

Filtering

☐ Create a filter on all-students view to only show students who are not registered to a campus

☐ Create a filter on the all-campuses view to only show campuses that do not have any registered students

Seeding & Pagination

☐ Seed 100+ students and 100+ campuses

☐ Implement *front-end* pagination for the all-students view (e.g. `/students?page=1` renders the first ten students, and `/students?page=2` renders students 11-20)

☐ Implement *front-end* pagination for the all-campuses view (e.g. `/campuses?page=1` renders the first ten campuses, and `/campuses?page=2` renders campuses 11-20)

☐ Implement *back-end* pagination for students (e.g. `/api/students?page=1` returns the first ten students' data, and `/api/students?page=2` returns students 11-20)

☐ Implement *back-end* pagination for campuses (e.g. `/api/campuses?page=1` returns the first ten campuses' data, and `/api/campuses?page=2` returns campuses 11-20)

Releases

No releases published

Packages

No packages published

Contributors   2

leafoflegend Eliot Szwajkowski

ThompsonHarris Thompson Harris