# Full stack CRUD Application - Platform for Managing Students and Campuses

* **Due**: June 14th, 2019 by 10:00AM
* **Points**: 25
* **Submitting:** GitHub URL to repository (front-end) and GitHub URL to repository (back-end)
* **Recommendation:** Use create-react-app to bootstrap the front-end portion of the application and use express-generator to bootstrap the back-end portion of the application
  + You might also need to look into setting up a `proxy` in the package.json of the front-end or the `cors` npm module
* If there is a time-crunch regarding setting up a back-end (database and routes), feel free to use dummy data on the front-end as a last resort
* **Group/Team Sizes**: 2 to 3

**Goal**

Using Node, Express, React, Redux, PostgreSQL, and Sequelize, build a RESTful full-stack web application to manage students and campuses. This will cover all of the CRUD operations such as Creation, Reading, Updating, and Deletion. This will encompass models, querying a database with an ORM, designing routes/endpoints and handler functions to process user requests and generate responses, writing out React Components, managing the state of the application with React-Redux, and much more. This will also involve having two individual repositories/applications (a separate server and a separate client), which encourages separation of concerns and modularity.

**Assignment**

Complete the following general User Stories:

As a User I:

* [ ] will land on a visually pleasing homepage by default, which allows navigation to view **all campuses** and **all students**
* can navigate to **all campuses view**, and
  + [ ] see a list of all campuses in the database
  + [ ] see an informative message if no campuses exist
  + [ ] add a new campus
    - [ ] with a validated form displaying real-time error messages
    - [ ] which redirects to the new campus’ **single campus view**
* can navigate to a **single campus view**, and
  + [ ] see details about a single campus, including enrolled students (if any)
  + [ ] see an informative message if no students are enrolled at that campus
  + [ ] navigate to any student’s **single student view**
  + [ ] delete the campus (and elegantly handle associated students)
  + [ ] edit campus information (including adding/removing students)
    - [ ] with a validated form displaying real-time error messages
    - [ ] which redirects back to the **single campus view**
* can navigate to **all students view**, and
  + [ ] see a list of all students in the database
  + [ ] see an informative message if no students exist
  + [ ] add a new student
    - [ ] with a validated form displaying real-time error messages
    - [ ] which redirects the new student’s **single student view**
* can navigate to a **single student view**, and
  + [ ] see details about a single student, including the campus at which they are enrolled (if exists)
  + [ ] see an informative message if student is not enrolled at a campus
  + [ ] navigate to **single campus view** of the student’s enrolled campus
  + [ ] delete the student
  + [ ] edit the student’s information (including campus s/he is enrolled at)
    - [ ] with a validated form displaying real-time error messages
    - [ ] which redirects back to the **single student view**

**Complete =** Completes all User Stories above

**Incomplete =** Does not complete all User Stories above; needs improvement or missing submission

**Here is a more technical breakdown/guideline of what’s expected from the above User Stories...**

**All Campuses and Students (21/57)**

**Backend (Sequelize)**  
- [ ] Write a `campuses` model with the following information:  
 - [ ] name - not empty or null  
 - [ ] imageUrl - with a default value  
 - [ ] address - not empty or null  
 - [ ] description - extremely large text

- [ ] Write a `students` model with the following information:  
 - [ ] firstName - not empty or null  
 - [ ] lastName - not empty or null  
 - [ ] email - not empty or null; must be a valid email  
 - [ ] imageUrl - with a default value  
 - [ ] gpa - decimal between 0.0 and 4.0

- [ ] Students may be associated with at most one campus. Likewise, campuses may be associated with many students

(**Express and Sequelize**)  
- [ ] Write a route to serve up all students  
- [ ] Write a route to serve up all campuses  
  
**Frontend (React-Redux, React, and React Router)**  
- [ ] Write a campuses sub-reducer to manage campuses in your Redux store  
- [ ] Write a students sub-reducer to manage students in your Redux store

- [ ] Write a component to display a list of all campuses (just their names and images)  
- [ ] Write a component to display a list of all students (just their names)  
- [ ] Display the all-campuses component when the url matches `/campuses`  
- [ ] Display the all-students component when the url matches `/students`  
- [ ] Add links to the navbar that can be used to navigate to the all-campuses view and the all-students view

**Single Student and Single Campus (14/57)**  
  
**Backend (Express and Sequelize)**  
- [ ] Write a route to serve up a single campus (based on its id), \_including that campuses' students\_  
- [ ] Write a route to serve up a single student (based on their id), \_including that student's campus\_  
  
**Frontend (React and React Router)**  
- [ ] Write a component to display a single campus with the following information:  
 - [ ] The campus's name, image, address and description  
 - [ ] A list of the names of all students in that campus (or a helpful message if it doesn't have any students)  
- [ ] Display the appropriate campus's info when the url matches `/campuses/:campusId`  
- [ ] Clicking on a campus from the all-campuses view should navigate to show that campus in the single-campus view  
  
- [ ] Write a component to display a single student with the following information:  
 - [ ] The student's full name, email, image, and gpa  
 - [ ] The name of their campus (or a helpful message if they don't have one)  
- [ ] Display the appropriate student when the url matches `/students/:studentId`  
- [ ] Clicking on a student from the all-students view should navigate to show that student in the single-student view  
  
- [ ] Clicking on the name of a student in the single-campus view should navigate to show that student in the single-student view  
- [ ] Clicking on the name of a campus in the single-student view should navigate to show that campus in the single-campus view  
  
**Adding a Campus and Adding a Student (12/57)**  
  
**Backend (Express and Sequelize)**  
- [ ] Write a route to add a new campus  
- [ ] Write a route to add a new student  
  
**Frontend (React and React Router)**  
- [ ] Write a component to display a form for adding a new campus that contains inputs for \_at least\_ the name and address.  
- [ ] Display this component EITHER as part of the all-campuses view, or as its own view  
- [ ] Submitting the form with a valid name/address should:  
 - [ ] Make an AJAX request that causes the new campus to be persisted in the database  
 - [ ] Add the new campus to the list of campuses without needing to refresh the page  
  
- [ ] Write a component to display a form for adding a new student that contains inputs for \_at least\_ first name, last name and email  
- [ ] Display this component EITHER as part of the all-students view, or as its own view  
- [ ] Submitting the form with a valid first name/last name/email should:  
 - [ ] Make an AJAX request that causes the new student to be persisted in the database  
 - [ ] Add the new student to the list of students without needing to refresh the page  
  
**Removing a Campus and Removing a Student (10/57)**  
  
**Backend (Express and Sequelize)**  
- [ ] Write a route to remove a campus (based on its id)  
- [ ] Write a route to remove a student (based on their id)  
  
**Frontend (React and Axios/Fetch)**  
- [ ] 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

**Total: - / 57**

The following 15 images (wireframes) can help guide your decisions regarding UI/UX:



























