Partners

Team Triceratops	Savannah	Antonio	
Team Allosaurus	Krista	Lucas	
Team Ankylosaurus	Claudia	Ken	
Team Iguanodon	Steve	Collin	
Team Brontosaurus	Michelle	Christian	
Team Dilophosaurus	Chris	Chad	Erica
Team Troodon	Adam	Angela	

Lot Bot

Parking is a pain, so users rejoice when the Lot Bot comes to town. Lot Bot keeps track of all of the parking spaces in a given lot, allows users to claim them, and then automatically bills them when the user departs. Many cities have fought to get the hot Lot Bot brought in.

Alas, Lot Bot Enterprises has a dirty secret: there is in fact no Lot Bot, and they need one as quickly as they can get one. They've enlisted your stalwart team to get the job done, and they're gonna need the help of some top-notch frontend and backend developers to not get caught in a Lot Bot thought knot. Their requirements are as follows:

The API

There are two key concepts that the entire Lot Bot system is based on: **lots** and **cars**. The API should support the following operations, and all should use JSON for encoding request and response bodies.

Method	URL	Description
GET	`/lots`	Get a list of all lots in the system, including # of spots.
GET	`/lots/<id>`	Get a list of the status of all spots in the specified lot, including the license plate # of anyone parked.
POST	`/lots/<id>`	Park a new car in the specified lot. Must send the `Car object` in the request body.
PUT	`/lots/<id>/<spot>`	Open up the specified spot and return the total owed.
GET	`/transactions`	Return a list of all transactions, along with the bill and license plate number of the charged vehicle.

The frontend

The frontend should make it easy for lot managers to record when drivers come and go, and should make it possible to monitor lots and track all expenses across all lots.

Requirements

- You should create three different views: a **lot list**, a **lot overview**, and a **transaction report**. You must use React and Redux and create a single-page app allowing you to navigate between all three.
- The **lot list** should show all parking lots registered in the system, as well as the number of occupied spots and total spots. All lots should be clickable.
- When a lot is clicked, the application should transition to the **lot overview** for that lot. The lot overview displays the lot name, # of occupied spots, and total # of spots. Additionally, you should render a visualization of the parking lot where occupied spots are red and unoccupied spots are green. When the user clicks a green spot, a modal dialog pops up and asks for the parker's license plate number and the spot then becomes red. When the user clicks a red spot, a modal dialog pops up and displays the total bill and the spot then becomes green.
    - You must create the modal dialogs yourself for this assignment and cannot use built-in or third party options!
- The final view is the **transaction report**, which lists all transactions that have occurred across all lots. Each transaction should list the license plate number and total cost of the transaction as well as when the transaction occurred.

The backend

For this project, your API is going to be responsible for responding to every request in the table above, and maintaining the state of the parking lots. You'll need the following models:

- Lot
    - id: int
    - spaces: Space[]
- Car
    - licensePlate: String
- Space
    - car: Car
- Transaction
    - car: Car
    - checkedInDate: Date
    - checkedOutDate: Date
    - price: double

When a car parks in a space, you will need to set the "car" field for that space and create a "transaction" object. When a car leaves a space, you'll find the transaction applicable to the car, set

the checkedOutDate of the transaction. At this time, you'll have to calculate how long the car spent in that space, and calculate the price that the person must pay, and free up the given space.

You will be given a list of lots to start with.

Hard mode

Come up with new features on your team and implement them.