Home Assignment - Full Stack Reservation App

The test assignment is meant to see how productive and autonomous you are by implementing a simple React/Next.js app.

You are free to design the page as minimalistic as you want, just use tailwind classes.

Problem Statement

Imagine a company that has a single shared car for all employees, each employee should reserve the car before using it. This web app will only run as a single instance on one machine at the company. People will come to that machine to reserve.

Rules of Reservation

1. You can reserve the car for a minimum of one hour, a maximum until the next workday at 11 AM
2. The workday starts at 9 AM and to 5 PM (don’t worry about the timezone, the local timezone will do)
3. Sunday and Saturday are not workdays (don’t account for the public holidays)
4. The duration of reservation in the workday should be a multiplication of one hour (e.g. 1 hour, 2 hours, 3 hours)
5. You should provide your name in the form of a reservation.
6. Your reservation should not collide with others.
7. You cannot reserve into the past!

Rules of implementing

1. Framework: Use React, Typescript, and Next.js (With Api routes). Tailwind for Styling.
2. Use Context if you need any global state. (you can use zustand, but do not use redux)
3. All reservations should be persisted using the Next.js API routes inside a postges DB
4. Use any npm package you like. (but don't bloat)
    a. you can use if you want this kind of packages
        i. `date-fns` for handling and formatting dates
        ii. `react-calendar` for the calendar
        iii. `react-hook-form` if you need to manage the state of the form.
        iv. `react-query` for managing server state (you can also use TRPC for end to end type safety and better DX)
        v. `next-auth` for managing user authentication

How the UI should look like (Calendar)

1. The application consists of One main route. in that route, you have the calendar for the current month, the current date reservations, and a reservation form (disabled by default)
2. The calendar should indicate the number of that day's reservation when we hover over it. also for each day, it should show a different shade of green (higher when there are more reservations)

3. The details
    a. Once the user clicks on the date he can see what users are and for how long they are using the car that day. (if any)
        i. format
            1. {name} - {from} → {to}
        ii. example
            1. User 1 - 10 AM → 11 AM
            2. User 2 - 12 PM → 2 PM
            3. User 3 - 4 PM → Monday At 11 AM
4. Once the user clicks on the date the form should be enabled (Unless there are no more spots left, in that case, the form should be disabled)
5. You should ask for the “from” and “to” in two select boxes, one for the start hours and one with the end of the reservation selection.
6. They will put the required fields :
    a. available time slots as dropdown
    b. submit button.
7. a user cannot have multiple reservations per day
8. and show a clear error or indicator of why the user can’t do an action
9. User should be able to logout

How the UI should look like (Rest of the pages)

- **Login page** (No need for anything fancy even the default next-auth will do)

General Guidance

1. Try to commit as atomic and as much as possible.
2. Push often on GitHub.
3. Try to turn on an auto-deploy with vercel.
    a. this is basically to receive feedback on the implementation if needed
4. Ask questions in the public group if there are any vague parts in the specification.

5. Take care about the file structure.
6. Abstract the complex logic in a custom hook or helper functions
7. Write unit - integration - e2e tests. (not needed but helps)
8. A readme for running the app should help