Next.js Exercise

Description

So, the exercise scenario goes, you've been contracted to create a Curbee competitor, "Carbee".

Carbee is operating on a limited budget, so there isn't an expectation that things will look great visually, but there is an expectation that the code will follow framework best practices. Carbee is serving a region that is limited to 3G cellular networks, thus application performance is important.

Also, they don't have a robust set of API's yet, and as a side effect, the requirements are limited, but can be summarized as:

1. Display a user's appointments on a dashboard
2. Display appointment availability for a given date
3. Oh, and allow users to log in

In this exercise, you'll be creating a web application using **Next.js**. You'll be leveraging existing API's to:

- Authenticate users
- Persist a session
- Authorize requests
- Retrieve data

The backend has a pre-filled database to support the tasks afoot.

You're welcome to use third-party libraries to assist you.

Getting Started

Create project

Create a Next.js project using the following instructions:
https://nextjs.org/docs/api-reference/create-next-app

Config

In order to hit our APIs, you'll need to add a proxy configuration to your next.config file. Without it, you'll run into CORS issues. You can find a solution here:
https://nextjs.org/docs/api-reference/next.config.js/rewrites

API Docs

Docs for the backend you'll be using can be found here:
https://modulith.herokuapp.com/webjars/swagger-ui/index.html

API Base URL

https://modulith.herokuapp.com/

Version Control

Create a repository in Github to push your code to. You'll use this to share your work with us. **When creating commits, be sure to use best practices, as if you're working with a team.**

Colors (optional)

In case you don't want to spend time putting together a color palette, here's a codepen with the Curbee palette in the form of CSS variables:
https://codepen.io/sodapop/pen/NWLmGqB

Specifications

Project

- Use typings, either typescript or JSDoc. What matters is that the typeflow can be properly evaluated/linted by a typescript server. Bonus points for practical usage of algebraic data types.
- Write code as if you're working with a team.

Authentication

Feeding the backend is a database pre-filled with the information you'll need to complete the required tasks – this information includes users. You can use the following credentials to log in as a pre-existing user:

> **Username: candidate@curbee.com**
> **Password: password**

For this task, you'll need to…
Create a user login form consisting of ***username*** and ***password*** fields

1. Use the login endpoint to authenticate a user

And on the heels of authentication, we have ...

Authorization

The backend authorization strategy leverages json web tokens. There are many ways of persisting a token and attaching it to request *authorization headers*. ***In this task***, I'd like you to demonstrate how *you'd* orchestrate ...

1. Token management/persistence
2. Request authorization, and how to handle it in an isomorphic environment
3. Auth-only pages/routes (protected routes)

Pages

For this task, I'll outline the requirements for each page that will exist in our app.

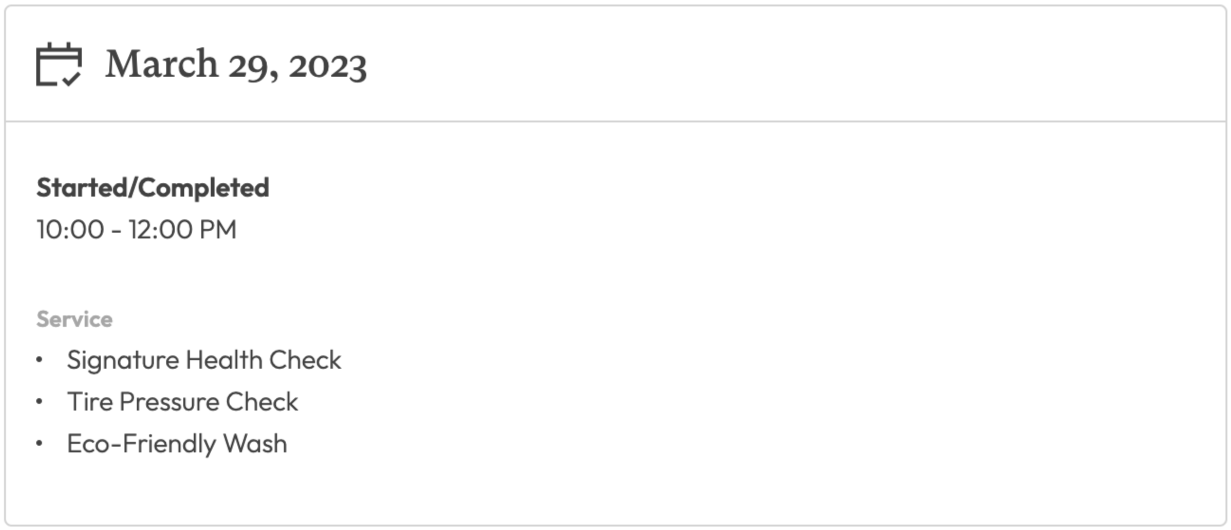
/login (public)

- Requires a login form
- Makes a request to the login endpoint
- Redirects user to `/dashboard` on success

/dashboard (protected)

- GETs availability and lists times by a selected upcoming date (eg: 2023-03-30), starting with tomorrow. You can use a <select> to choose availability, or get creative with it if you so choose.
- GETs and displays user's appointments
    - This endpoint supports pagination via query parameters. Add pagination controls to the UI to retrieve data for previous and next pages.
    - Create a UI component to display the appointment's…
        - Appointment status (Scheduled, In-Progress, etc)
        - Start time startTime
        - Appointment duration
        - CompleteTime in 12 hour time.
            - Note that complete time may not be populated if the Appointment has not been completed
        - Service (workOrderDto.service)

The design shown below isn't required and doesn't reflect exactly what your payload will look like, but feel free to use it as a guide so as not to spend too much time brainstorming your own:



March 29, 2023

Started/Completed
10:00 - 12:00 PM

Service
- Signature Health Check
- Tire Pressure Check
- Eco-Friendly Wash

Project Reflection

1. Did you run into any “gotchas” along the way? If so, what were they and how did you address them?
2. How did you handle forms? In a largely form-driven project, would you do anything differently? If so, what?
3. How did you handle authorization? In your ideal FE/BE scenario, what auth strategy would you use?
4. Is there anything you’d like to share about your project prior to my evaluating it?
5. How long did you spend on this exercise? If you had unlimited more time to spend on this, how would you spend it and how would you prioritize each item?