User interface

1. *Provide a short description of what the user interface you design might look like?*

First interaction.
When the user opens the app, the user sees a small landing page with a bit of a techy background image/carousel, and a login/signup call-to-action button. Once the user signs in, the app redirects to user dashboard. If the app is not meant for public use, landing does not make much sense. In this case, the user first sees the login/signup page.

Dashboard.
User dashboard includes all necessary pages to view and reserve available workstations. Pages in the dash would be: “list of available workstations”, “my reservations”.

Dashboard Menu.
Menu is always visible when navigating between pages in the dash. On desktop, dashboard menu can be positioned on the left side while being always visible. On mobile, the menu can be hidden behind the hamburger icon. It would slide into view when the icon is clicked. On desktop, however, the menu might waste a lot of free space due to the small number of pages. If the number of dashboard pages will not increase, it makes sense to just use tabs on top of the layout instead of a left-side menu.

List of available workstations dash page.
This subpage has a grid view of cards which represent workstations. Each workstation card has a badge or dot which reflects its status: available or not available. Availability times can be shown in a popup/modal when the user clicks on the “Check availability” button. I think having a separate page for each workstation with 2-3 data fields is an overkill.

My reservations dash page.
This page can come in with a grid view of cards, too. Nothing special here. If there were requirements for more complex reservation management, I would have added action buttons directly on the cards.

2. *Please write a React component for a workstation, that displays its id, desk number, current reserved status, and a schedule showing when it is reserved and available. Assume that the schedule component is provided by another React component. Assume the Workstation component fetches its own reserved status and schedule on mounting. Feel free to split up the component into multiple components in order to make them as clean and reusable as possible.*

Please find source code for relevant components at https://github.com/shelooks16/workstation

Live preview is available at https://shelooks16.github.io/workstation/

I decided to not use any external solution or a library except *dayjs* to deal with date-times. The API is mocked and intentionally timeout'd. Styles for components are written with plain CSS (CSS modules).

3. *Current reserved status is displayed with either a green dot (currently unreserved) or a red dot (currently reserved). Please include this in #2 above and provide the CSS for the dots.*

Please find a reusable StatusDot component at
https://github.com/shelooks16/workstation/tree/master/src/StatusDot

REST API

4. What REST endpoint Method and URI would you put in place to get a Workstations schedule? Include any query params or request body.

To retrieve a Workstation's schedule I would certainly use a **GET** endpoint which, for instance, can be defined as **/workstation/:id/schedule**, where **:id** is a dynamic segment representing workstation id for which to get the schedule.

Alternatively, depending on the style of the REST API, the endpoint might look different. For example:

Method: **GET**
Pathname and params: **/workstation_schedule?workstationId=123**

where **workstationId** is a query parameter representing workstation id for which to get the schedule.

5. What REST endpoint Method and URI would you put in place to reserve a Workstation for a specific user? Include any query params or request body data you think it might need.

To reserve a workstation for a specific user, the endpoint might look like this:

Method: **POST**
Pathname: **/workstation/:id/reserve**
Body: **{**
‘userId’: ‘123’,
‘from’: { ‘date’: ‘2023-27-10’, ‘time’: ‘15:00’ },
‘to’: { ‘date’: ‘2023-27-10’, ‘time’: ‘15:30’ }
}

where dynamic segment **:id** identifies the workstation again, **userId** included in the POST body represents the user for which the workstation must be reserved, **from** and **to** fields tell reservation time range. This endpoint might be dangerous if not protected by proper authorization rules. Without authorization, anybody can use this endpoint to reserve workstations for other users in the system. If the API allows users to reserve computers for other users, the endpoint should enforce authorization checks for the user who makes the request. Otherwise, the endpoint should allow the user to reserve only for himself (after authentication, of course).

Data Persistence

6. What might be the different types of data you would need to store and access for this app?

This type of system needs a database to store user records, list of workstations, reservations.

I don’t think any file storage will be needed unless there is a specific requirement.

Other

7. What questions would you ask the PM providing these requirements?

- How will workstations be imported into the system?
- Workstations sound like they need some sort of administration. What if a workstation gets damaged? What happens next? How will the workstation be temporarily disabled or removed from the system? Does the system need to introduce admin accounts or have a separate admin panel?
- Do we take timezones into consideration?

- Should each workstation have its own dedicated page?
- Do we need to collect data from the web-app to gain insights later on? Do we need to integrate something like GTM?
- Is there only one shared lab or many?
- How many workstations can the user see at once?
- In case of a huge workstations list, do we need to implement filter-based search? Pagination, infinite scroll?
- Can the 30-minute time chunk requirement be potentially changed in the future? Let’s say instead of a 30-minute chunk, users can be allowed to book for 2 hours straight.
- Will we need to send notifications to the users about upcoming and ending reservations? For instance, push notifications.