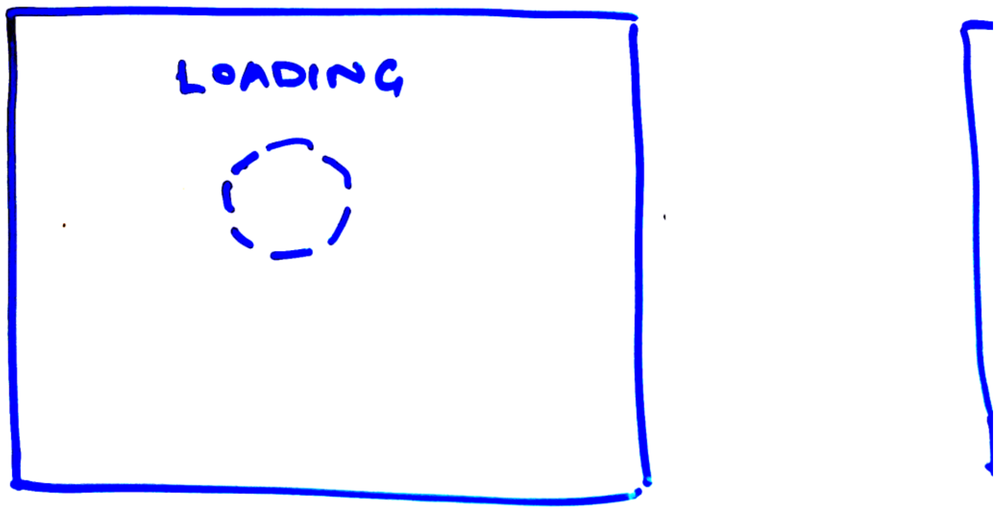
User Profiles

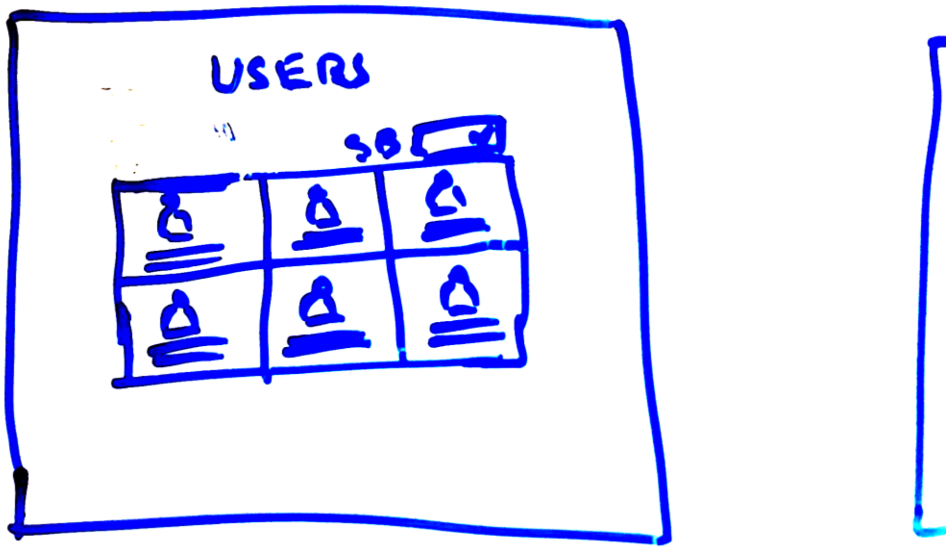
Step 1:

- Get user data from this API: https://reqres.in/api/users?delay=3
- If you need detailed API documentation, visit https://reqres.in
- Show a progress spinner while waiting for the data, refer wireframe below:



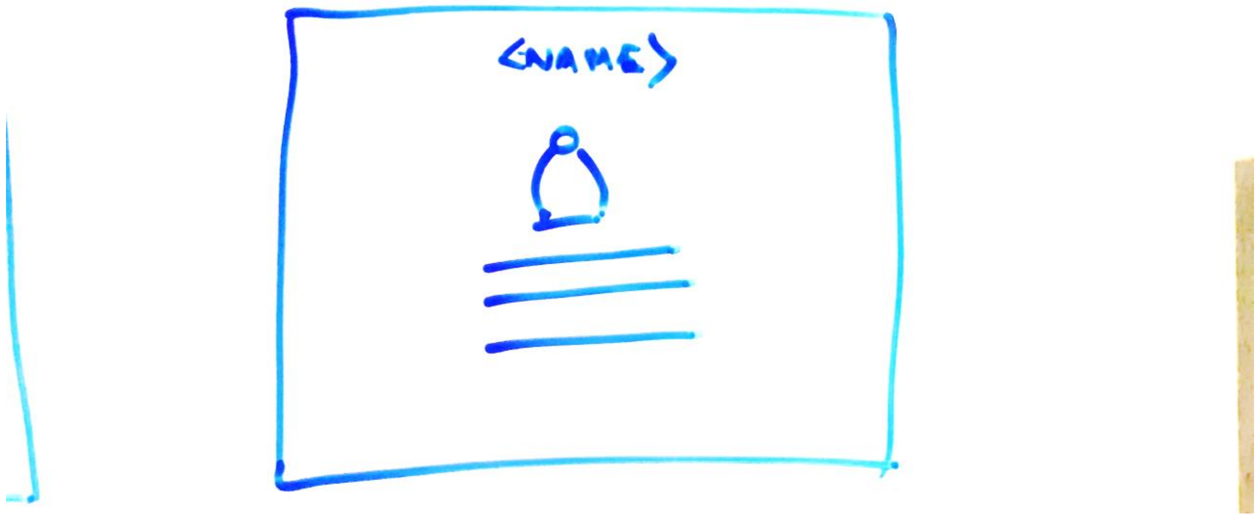
Step 2:

- Display users in the form of a tile grid.
- Each tile should contain the image of the user, first name and last name.
- The URLs for user images are available in the data itself.
- These tiles should be clickable.
- There should be a sort by dropdown above the grid on the right.
- Refer wireframe below:



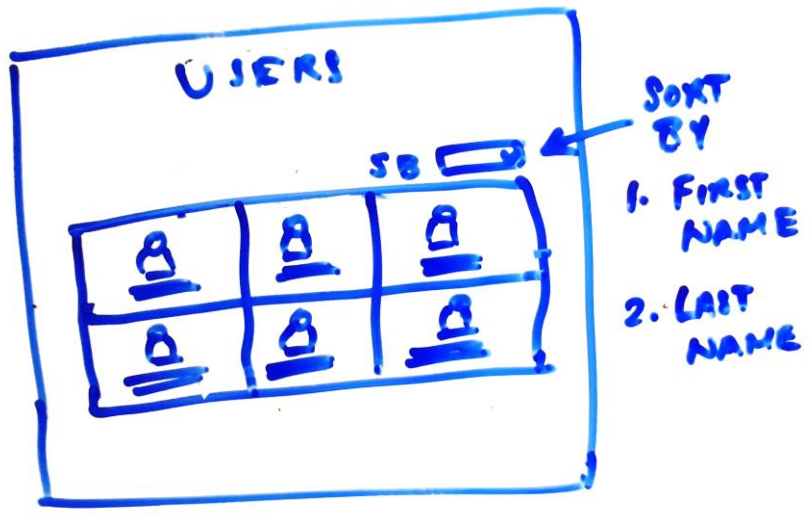
Step 3:

- On clicking on a tile, the profile page of the user should be shown. The API for this is https://reqres.in/api/users/<id>
- This page should contain a large image of the user and the rest of the details. Also handle empty state – if no user with such id exists.
- Refer wireframe below



Step 4:

- There should be sort options in the main users' page. The available options are: **None, First Name and Last Name**
- On selecting one of these, the tiles should sort and rearrange.
- Refer wireframe below:



- **The final product should be hosted somewhere (check reference links) and the URL should be shared**

- **Code should be committed to a GitHub repo and the URL should be shared.**

Requirements:

- Use Vue – Vue2 or Vue3. Use vue-cli/vite or alternatives.
- Do NOT use frameworks like Nuxt
- Use any UI library of your choice or write your own CSS/SASS. Please make sure it's presentable on popular screen sizes.
- Use state management library to handle data.
- Use configuration files wherever necessary - constants, shared variables over the project.

Evaluation Criteria:

- README.md for project and setup details
- Functionality of the project – meet all the steps and requirements.
- Modularity of code
- Test cases – unit tests
- Use of state management
- Transitions (loading to data, users to profile page, sorting, etc.)
- Git commits – messages and code state
- Consistency in linting – ES Lint, Prettier.
- Optimization in data fetching and renders.

Extras:

- API has more pages for users, implement pagination for that on users' list page. Hint: read docs, pass page as a query
- Use commit hooks to block git commit if the code does not pass lint check – hint: use husky, lint-staged.
- Implement CI or CD with any tools of your choice – GitHub Actions, Travis, CircleCI, for example.
- Implement a mechanism to measure test coverage (with a command such as npm run coverage)
- Reduced bundle size – implement tree shaking, code splitting if you find bundle is large

If you believe you need to document something such as your analysis, findings, choices you made vs alternatives, add those to readme.

Reference:

UI Library:
Tailwind
Bootstrap
Material UI

Development:
Vue: https://vuejs.org/
Vuex: https://vuex.vuejs.org/
Pinia: https://pinia.vuejs.org/

Deployment:
https://www.netlify.com/
https://vercel.com/
https://fly.io/
https://render.com/
https://pages.github.com/