1. AOE Card Game

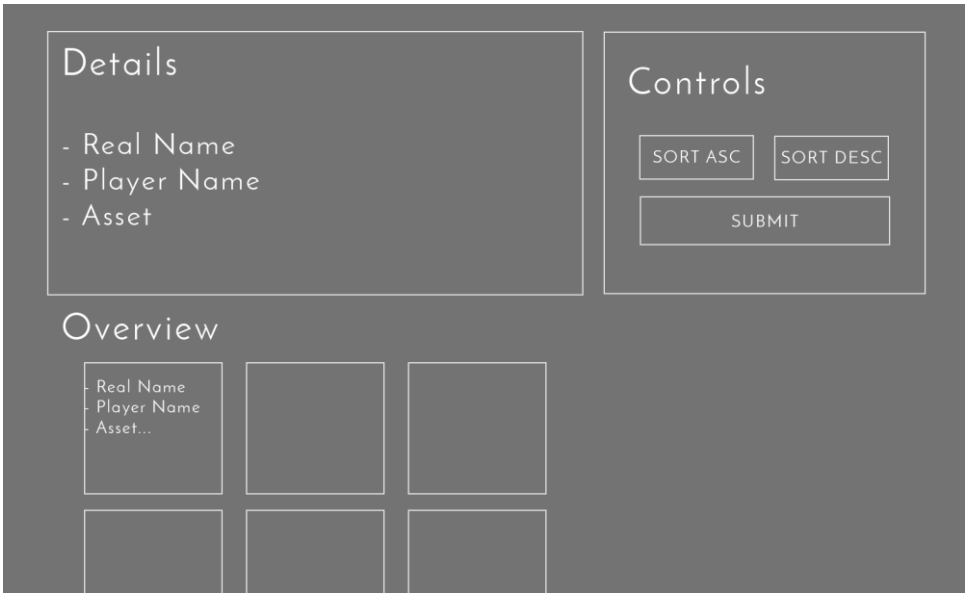
1.1. The Story

In an agency far away a PO had a product vision.

Because her dev skills are limited, she immediately turns over to you and asks you to develop a MVP, which shall feature her vision.

Her vision is a card presentation. She provides you with some text and a very hasty first draft (see the screenshot below), which probably needs some changes but highlights her vision:

Small cards display information about players that participate in a card game.



1.2. Card data

One of the Backend Devs provided a mocked backend where you can already fetch the data from with a GET request: https://opensource.aoe.com/the-card-game-data/player.json. Obviously this is just a json file, but treat it as if it were the real api.

1.3. Requirements

The game shall be a single page application (SPA) where all actions and visualizations are shown on one page and will be implemented in modern JavaScript/Typescript. The MVP actually targets only desktop environments with the latest browser generation.

1.3.1. Views

1.3.1.1. Overview

The PO thinks that there shall be only three cards per row. Also she thinks there should be no line breaks for the text within a card in the overview. Instead there shall be three dots (...) to indicate, that the information is larger than the horizontal space available. Only the large card (details view) shall show the full information even with line breaks if not enough space is available. Once a card from overview was selected, it's content is entirely shown in the details view while the selected card becomes highlighted to indicate what is currently active in the details view. It is also necessary to have some indication for users that shows that interactions are possible.

1.3.1.2. Details

Initially when nothing is selected the details view is hidden. It's important for the PO to avoid side effects like a sudden change of the layout when a card item gets selected and the details view is shown.

1.3.1.3. Controls and filter

There is one more section which holds all the UI elements for user interaction explained here (see functionality)

1.3.2. Functionality

1.3.2.1. Sorting

There should be functionality to sort the cards by real name in ascending and descending direction. Initially when the app is bootstrapped the cards should be shown as they come from JSON response (same order as in the table).

1.3.2.2. Submitting

There is also functionality required to submit the currently selected data from details view using a mocked AJAX request. The backend for this does not exist yet, so it is up to you to define a sensible JSON data structure and prepare everything for when the backend will be there.

1.3.3. Implementation

The last information you heard from the PO was, that you will implement an MVP now and that your colleagues will take over this piece of code in the future without you to extend it in another iteration.

After a short meeting you agreed with your colleagues on the following action points:

1.3.3.1. Must Haves

- Build out a basic modular folder structure
- asynchronously load the player data to be displayed as cards in JSON format
- Build an object model for the player data
- Provide the required logical views as in the explained image

1.3.3.2. Furthermore, you follow best practices like

- Separate concerns
- Avoid globals
- Use generic naming for both JS and CSS
- Use semantic HTML5

1.3.3.3. Things we usually do in our projects

- unit, integration and E2E tests
- static code analysis
- typescript
- using React, Angular or Vue
- create a production build while maintaining the development code
- and/or generally feel free to provide a setup which is positive (from your understanding) for the upcoming iterations when others take over the project and continue working on your implementation.
- documenting the project specifics to allow others an easier start into the project

2. Challenge specifics

Beside all the requirements above we do want you to develop the game in a manner where you would like to continue working on the code — *without the sentence "we need a complete rewrite ;-)"*.

While we will welcome any framework or library that supports you in delivering the desired outcomes, we (the AOE-frontend devs) are most comfortable in giving feedback in React, Angular and Vue.

With this challenge, there is **absolutely nothing to do on the backend**, simply focus on the frontend.

2.1. Focus

Depending on your personal skills, you may focus more on one of the two following aspects. Please do know that our expectations are higher

2.1.1. More Frontend less Javascript

If you're more focused on perfect markup, great styling and amazing usability etc. feel free to differ from the above mock in terms of layout, animation, responsivity etc. Keep the logical views such as overview, details and controlls/filter and provide us with an unique UI/UX. We're exited what you'll provide.

2.1.2. More JavaScript less Frontend

If you're more focused on JavaScript (software) development, feel free to use the mocked "design" above and provide a great piece of software. We're super exited and looking forward to get impressed by your solution.

2.2. Deliverables

Please provide your implementation either as an archive via mail or preferably via github (private repository, shared with: code-review-frontend@aoe.com).

Alternatively we accept a zip file with the project.

Please provide us a **short** overview of important decisions you took and why you took them in any format you like. You can either add them to your github or send them via mail to your contact person.

3. Conclusion

Feel always free to ask if things aren't clear enough — *we'll always have an open ear and will improve this test by our applicants feedback*.

Now it's time to start coding and we (the AOE-frontend devs) hope you will enjoy this challenge :)