Velope OTT Developer Remote Test

To	
From	
Date	

Prerequisites:

1. Go to The Movie Database website and register for an API key (https://developers.themoviedb.org/3/getting-started/introduction).
2. Bootstrap a JS project with any desired JS framework. This could be DOM based (eg. React Hooks, Preact, Svelte, SolidJS etc) or WebGL based (eg. PixiJS, AnimeJS, etc.) (the specific language/framework may vary according to the position you are applying for).
3. Typescript is optional, but preferred when it's available.

App Requirements:

Create an application that allows an user to browse a collection of assets (movies, shows) etc. These should be presented in the form of carousels, that the user can navigate with the arrows keys of the keyboard (Up and Down to switch carousels, Left and Right to scroll through the carousels, Enter and Back to navigate in/out of the details screen).

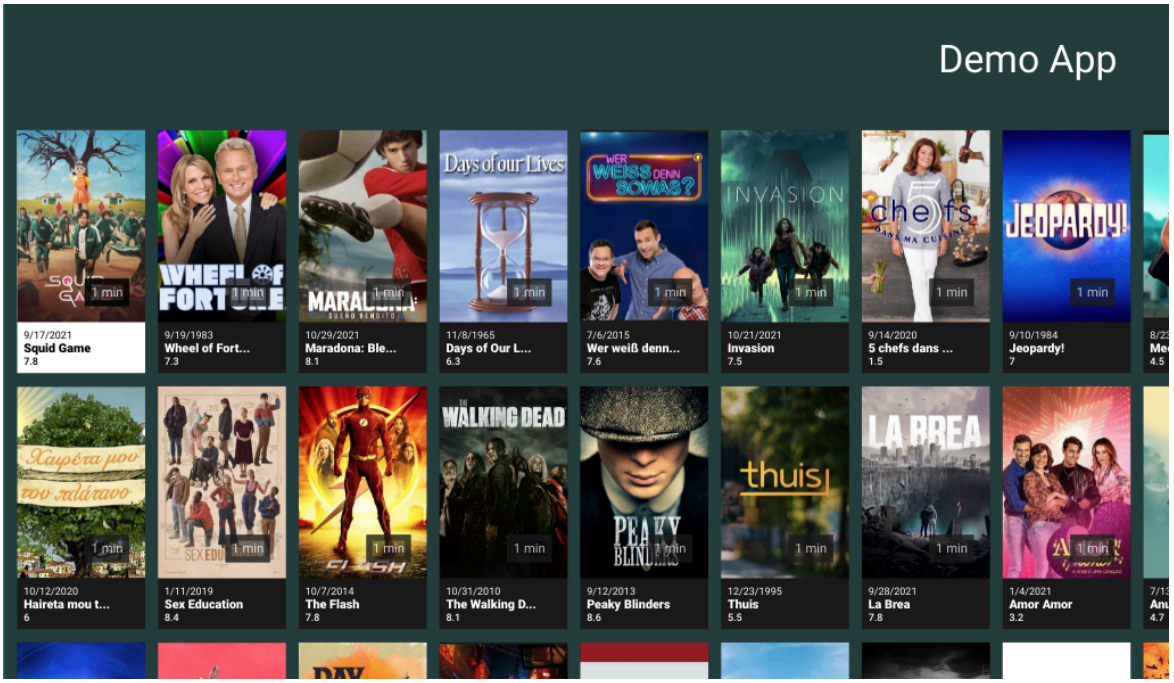
The user has a top navigation bar, where they can switch between 4-5 different genres (the first genre should be "All"). Switching to a specific genre should update the assets accordingly.

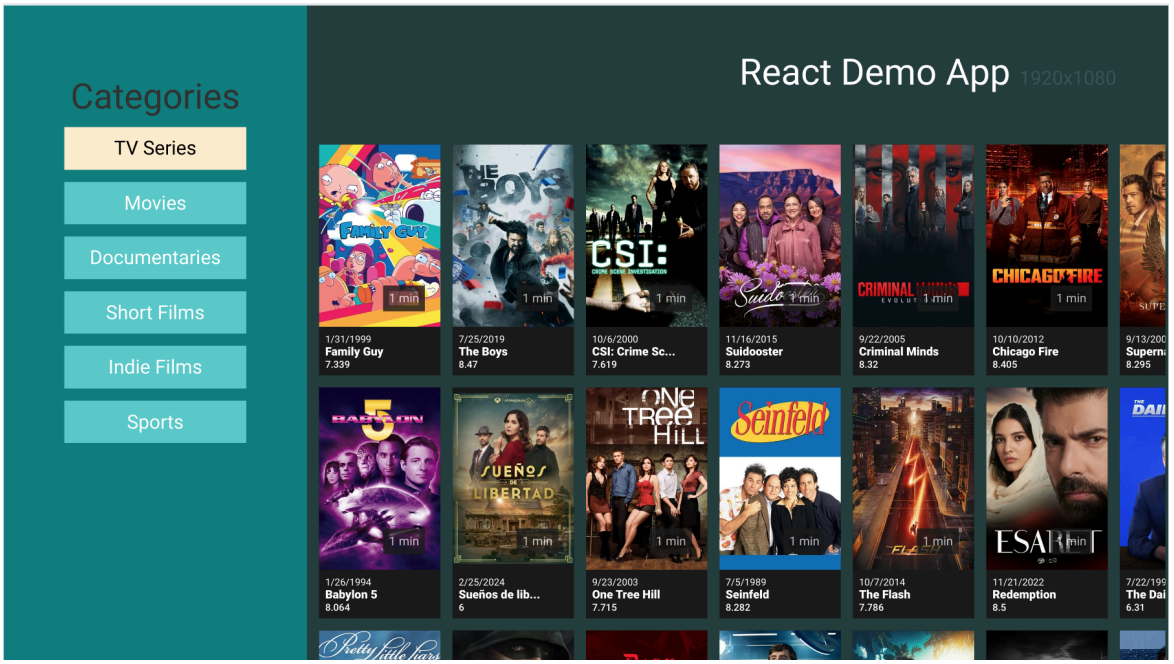
If the user selects an asset, the application will go to a new details screen for it, where the user can either read information about it or go back to the main navigation.

The app needs to display at 16:9 aspect ratio, same as a TV screen. Think of AppleTV, Netflix or Youtube (on SmartTVs) when thinking about the desired UX. The app will need to properly scale for 1280x720 and 1920x1080 resolutions at load time. (doesn't have to do this dynamically at run time, but when the user opens the app, it should be correctly scaled for both resolutions)

We recommend using your browser's device emulation feature to emulate a TV screen (for example, on Chrome https://developer.chrome.com/docs/devtools/device-mode/)

Example App UI:

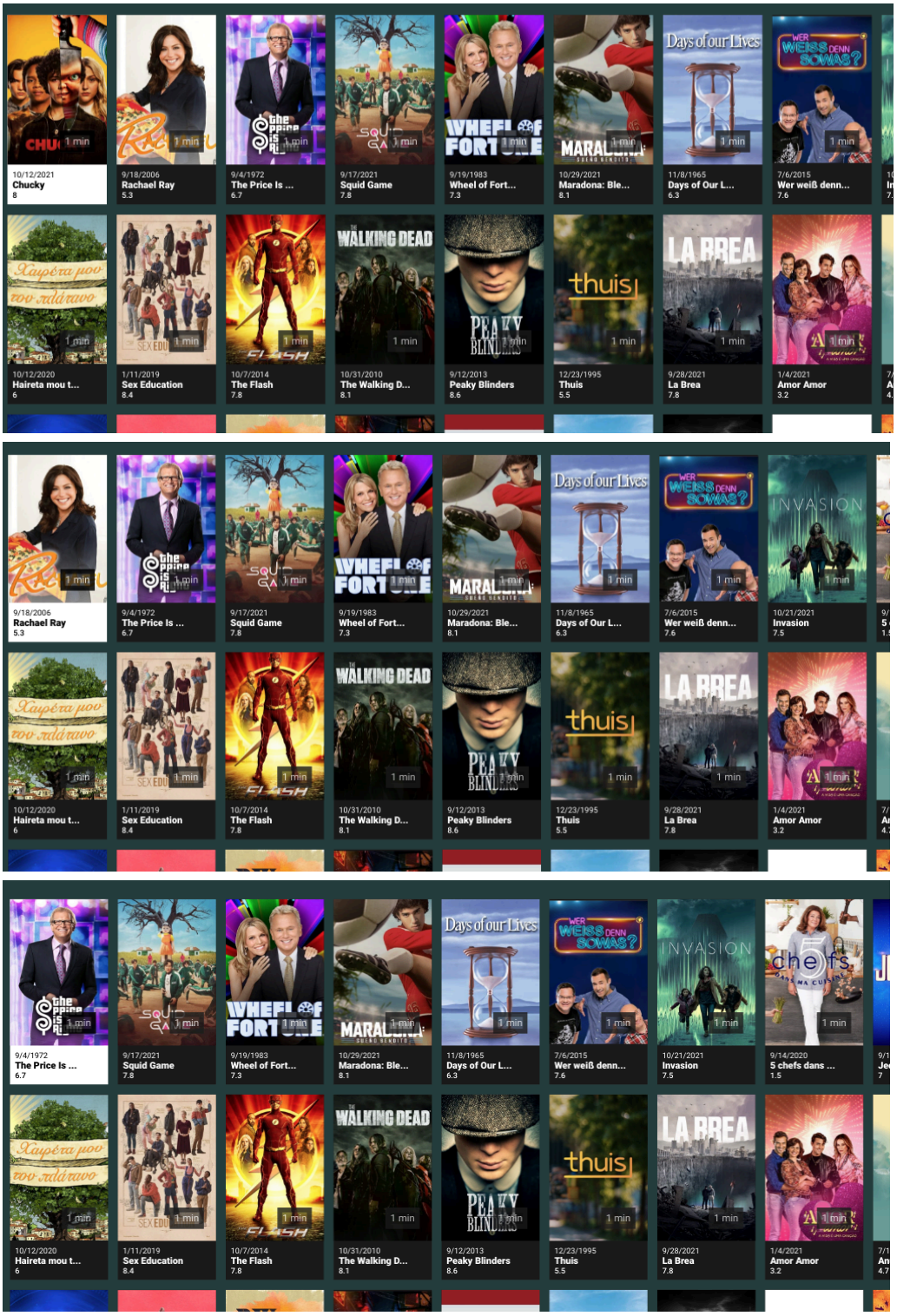




Developers are encouraged to improve on the example design as long as the acceptance criteria is still met.

Acceptance Criteria:

1. Navigation of the app must be fully implemented with keyboard buttons: the arrow keys, Enter and Back. Any test that can't be fully navigated using only those keys will be rejected.
    a. Bonus points: Mouse/pointer support in addition to keyboard keys support, custom mouse pointer
2. When the user first lands on the app, they should see either the carousels with the content, or a loading indicator until these are ready. Focus should be on the first item of the top navigation menu ("All"), which should be present from the start.
    a. Focus on each item, both on content and on the navigation menu, should be visible to the user. This can take the form of changing the base color of the item.
    b. Bonus points: A loading splash with an animated activity indicator that does not "hang" when the app is loading data and media. (Hint: CSS animations are not CPU bound)
3. The "content" (the assets carousels) should be made of carousels of at least 30 items per row, and at least 10 rows. Each item should contain at least a poster image, title and year.
    a. When the end of a list is reached, focus should roll back to the first item.
    b. When moving across the carousels the user should be able to leave the button pressed to move faster through the items. This continuous movement should feel fast.
    c. Bonus points: "Infinite scrolling" or "wrapping" of the carousels.
    d. Bonus points: on demand smart and lazy loading of the lists and rows with no perception that some things are preloaded just before being displayed
4. As the user moves through the carousels, the focused item or "cursor" should always stay on screen. Examples (notice the focused item, marked as white, stays in sight as the user moves right):



5. The user can navigate from the “content” (the asset carousels) to the nav menu and back.
6. The user can navigate between the items on the navigation menu, and when selecting them, the content should change accordingly.
7. The user can select any asset and, when doing so, is redirected to an item details screen, where they can see information about the asset. This page should at least show the asset poster, title, year, description. The user can hit “back” to go back to the previous page (it should keep the same genre they were navigating before).

   a. Optional: Putting a small mock UI in this screen with buttons like “Add to favourites” or “Play now”
8. Special care should be taken around movement speed - SmartTVs and connected devices are very low resource when compared to a browser, so animations should be as fluid as possible but also snappy so they don’t take a lot of time

Each of the previous points will be evaluated individually. Further feedback will be given after a code and app review.

Deliverables:

- Link to a public Github Repo of the app
- A link to a cloud deployment of the app running (eg. a heroku/vercel/etc URL with the app running).