2021

Full stack developer

Interview task

1 - Introduction

The test is divided in 2 tasks:

- The UI task will require you to adjust the layout accordingly to the requested design changes and you will replace the current class-based CSS with a more reusable grid component.

- The Node task will require you to create some endpoints using express and you will be processing some data using the REST or GraphQL version of the SpaceX public API in order to output the requested data as a list of cards.

2 – UI task Requirements

- The current grid is implemented using some CSS applied using a class on a div in the traditional way (see App.js). The task requirement is to remove the current CSS implementation and build a reusable grid component to place it in the 'layout' folder with the other layout components.

  NOTE: The complexity of this component is up to you. We understand time constraints, so we will consider the requirement fulfilled even if the component is made in its simplest form.

- The card component UI currently has an image at the top and the rest of the content is at the bottom of the card. We would like you to flip this layout, while keeping all the card elements of the same height. See attached image for visual reference. No change to the current HTML structure or order is allowed in order to achieve this task.

- Add simple type checks on components where you feel it's required.

2.1 – UI task bonus points

- Align the grid with the top navigation.

- Write a test for the launch card component.

2.2 – UI task technical constraint

- Grid and item alignment must be achieved without using the new CSS Grid specification.

- The CSS should be implemented using only Styled Components.

- No CSS plugin or library is allowed to fulfil the requirements. You should be able to find all you need in the current codebase.

3 – NODE task requirements

- In the repository you will find a simple server.js file. Please create the following endpoints using express:

  - ` **/launches** ` will return an array of 50 past launches including name, details, mission patch, and Youtube link.

  - ` **/rockets** ` will return and array all the rockets including name, description, cost per launch (formatted), picture of the rocket.

NOTE: You can reuse and extend the current launch card to output the data as a list of cards or create a new card specific for the rockets data.

3.1 – NODE task bonus points

- Show all the launches instead of 50 but with pagination
- Create a search input with a submit button so the user will be able to search the launches by name. (No autocompletion needed)
- Instead of using the provided buttons, create 2 different routes accessible from the main navigation with react router that will show either the launches list or the rockets list.

3.2 – NODE task technical constraint

- For this task you can use either the SpaceX REST API or the GraphQL version of it. What is important is that the endpoints will be server side handled and the requests made to either APIs will be done using node and express. (E.g., react Apollo client is not allowed but you can use apollo in express to work with the data coming from the GraphQL API).