Front-end Developer Assessment

Hello. 🙂
Thank you for applying for the Front-end position at CLO Virtual Fashion. Before proceeding to the interview, we are conducting a task phase to assess the candidate's capabilities that may not be visible from the resume.

The task is to implement the content list and filtering functionality for the **CLO-SET CONNECT Store Page** (https://connect.clo-set.com/store). Please follow the requirements outlined below.

Task Instructions

- **Duration: 1 week**
  You have one week from the day you receive this notice to implement and submit your solution via email.
  (e.g., If you receive the email on Thursday, the deadline will be by 11:59 PM the following Thursday.)
- **Submission**: **GitHub public repository URL**
- If you have any questions during the task, please send them via email to recruit@clo3d.com.

Conditions

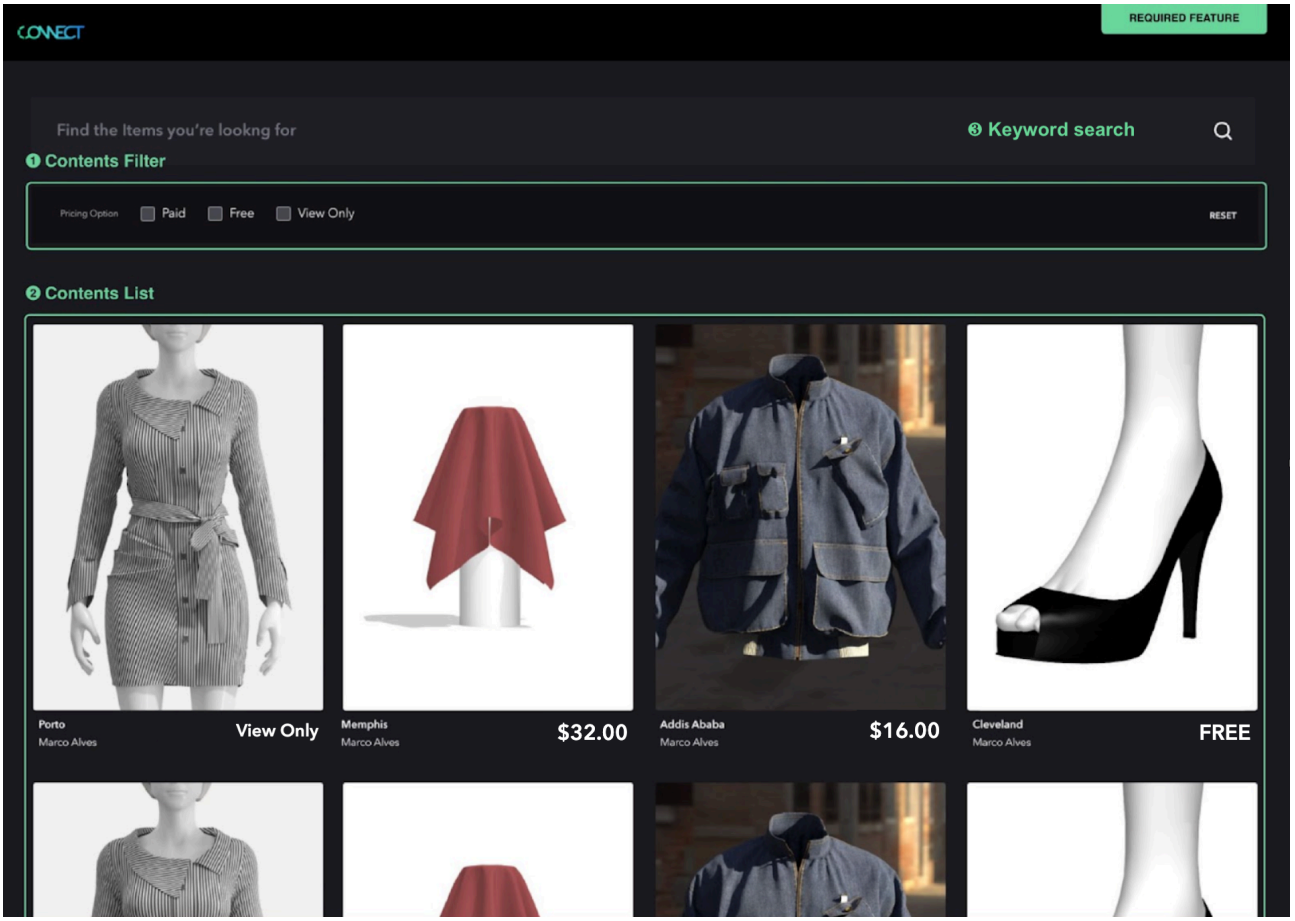
- The project should be developed using **React.js**.
- Use a **state management library** (e.g., Redux, Recoil, etc.).
- You are free to use **CSS-in-JS** or a **CSS pre-processor**, and any **test code libraries** of your choice.
- The project must be runnable in a local environment.
  (e.g., `npm run start` should allow you to access the app on `localhost:3000`.)
- Use **Git** for version control.
- You must use the provided **API** (data).

- The implementation should resemble the given design as closely as possible.
- There are no restrictions on the packages you can use, but you must explain why you chose each one.

Requirements

The requirements are divided into **Required** and **Optional**. You must implement all **Required** items, while **Optional** items are not mandatory but would be nice to have if time allows.

👇 Required

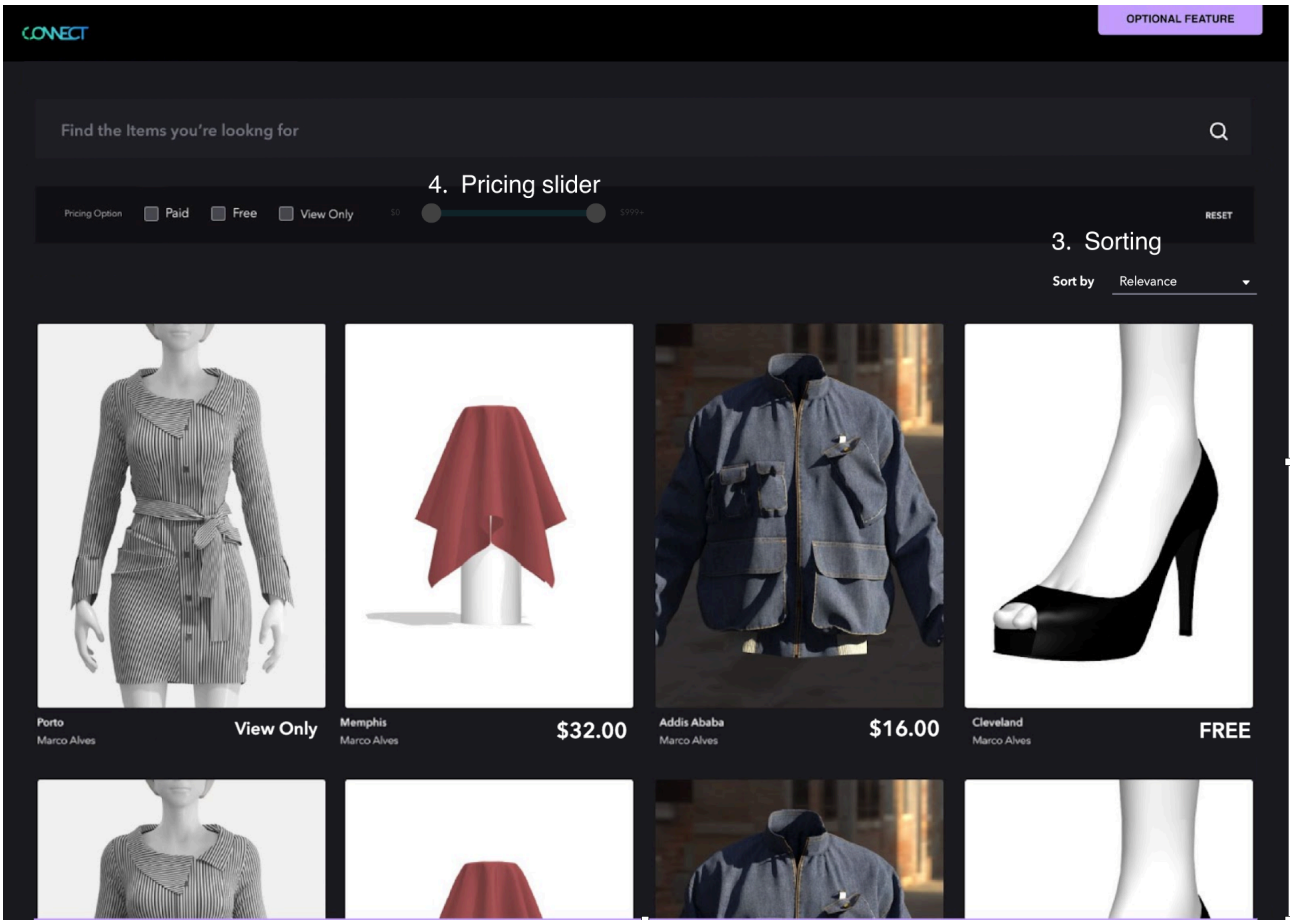


1. Contents Filter
    - Pricing Option
        - There are three options: Paid, Free, and View Only

        - The default state should be unchecked, and when all options are unchecked, all data (Content List) should be displayed.
        - Content List should be filtered based on the selected Pricing Option(s). (e.g., If Paid is selected, only Paid content should be shown.)
        - Multiple Pricing Options can be selected at once.
    - Clicking the **Reset Button** should restore the default state.
    - The filter or search results should persist across page reloads, but avoid using browser storage for this.
2. Contents List
    - Display each item's photo, user name, title, and the **Pricing Option** (Free/View Only), and for Paid items, the price should be shown.
    - Apply a **grid system** that adjusts based on the device width:
        - Default: 4 columns
        - Below 1200px: 3 columns
        - Below 768px: 2 columns
        - Below 480px: 1 column
    - Implement **infinite scroll** to load more items as the user scrolls (loading size can be determined as needed).
3. Keyword Search
        - Filter the list based on a keyword search (e.g., searching "Anisha" should filter content that includes "Anisha" in the user name or title).
        - If no keyword is entered, all items should be displayed.
        - Keyword searches should be combinable with Pricing Option filters. (e.g., searching for "Anisha" and selecting Paid should filter content that is both Paid and related to "Anisha.")
    - As with filters, the search results should persist across page reloads without relying on browser storage.

👇 Optional

1. **Test Code Writing**: Please include test code for your implementation.
2. **TypeScript Application**: The project should be implemented using TypeScript.



3. **Sorting Implementation**:
    - Implement a sorting feature in a dropdown format with the following criteria (sort only according to the criteria below, and there's no need to consider secondary sorting for items with the same value):
        - Item Name (Default)
        - Higher Price
        - Lower Price
4. **Pricing Slider Implementation**:
    - Create a range slider with a minimum value of 0 and a maximum value of 999.
    - The slider should be activated/deactivated based on whether the Paid option is selected or not.
    - When the handle is dragged and dropped, the selected amount should be displayed on both sides of the slider, filtering items only within that price range (the handles should not overlap).
5. **Skeleton UI for Infinite Scroll**: Implement a skeleton UI that corresponds to the infinite scroll of the content list.

Resource

👇 Data

- Pricing Option enum

```
enum PricingOption {
  PAID = 0,
  FREE = 1,
  VIEW_ONLY = 2,
}
```

- Contents List API: [GET] https://closet-recruiting-api.azurewebsites.net/api/data
    - Please use the API for data fetching, and implement the filtering logic directly on the frontend.

👇 Design

- Please develop the project to resemble the image below.
- Only the design elements required in the must-have and optional requirements will be taken into account for evaluation.