The goal is to develop single-page application with the following pages:

Dashboard page

URL: /dashboard

The page consists of header and a set of cards as shown above.
Each card represents a campaign. There are 3 types of campaign: Influencer, Challenge and Giveaway.
When **Dashboard** button (1) is clicked, the app redirects to this page.
When **Hide** button (2) is clicked, the corresponding card is removed from the page.
When **Show details** button (3) is clicked, the app redirects to the Details page (see details below).
When **Show more** button (4) is clicked, display 8 more cards. Hide this button if all cards are already displayed.

Dashboard mock data is available here.

Details Page

URL: /campaigns/<id>

Details page consists of header (same as in the previous page), campaign name, and tabs (Pending, Approved and Rejected).
The **social network** filter (5) is common for all tabs, and when the social network is chosen, posts are filtered accordingly.
In each tab we have posts that belong to the campaign. Each post is represented by card and can have one of 3 states: pending, approved, and rejected. When user hovers over post picture, post caption is shown. `

Pending tab.
Here we have **keyword** filter (6) (chips element in ngMaterial). When **Filter** button (7) is clicked, pending posts are filtered by caption.
When **Reject** or **Approve** button (8) is clicked, move post to Rejected/Approved tab accordingly.

@anamaddock
Instagram

LAST PHOTO WITH BLACK HAIR
#THINKPINK

LAST PHOTO WITH BLACK HAIR
#THINKPINK

LIKES	COMMENTS
2345	234

REJECT   APPROVE

@anamaddock

@anamaddock

LAST PHOT
#TH

LIKES
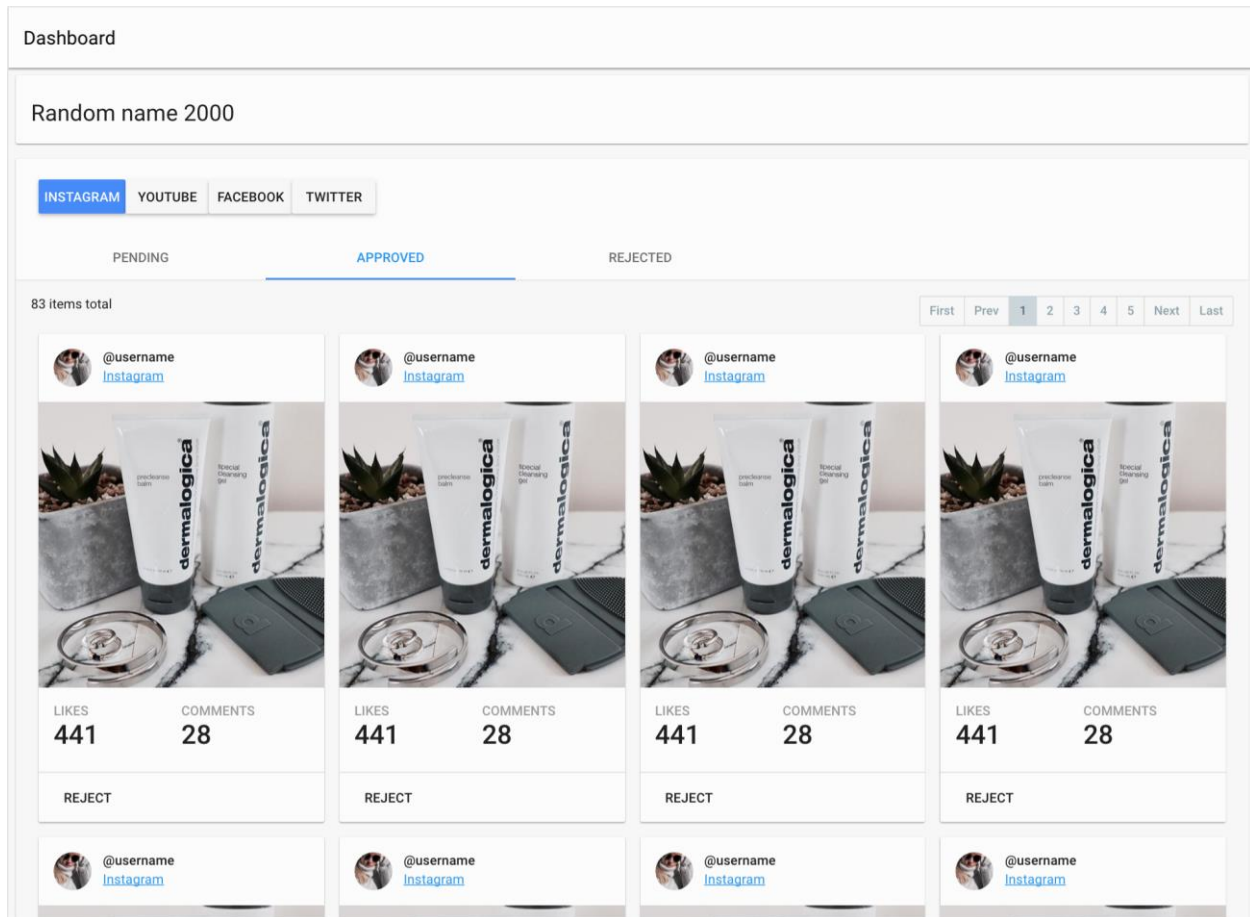2345

REJEC

(post caption when hovered over)

Approved tab looks similar to Pending tab, except there's no **Keywords** filter and **Approve**

button. Same for Rejected tab (but obviously post card has **Reject** button instead of Approve). Posts mock data is available here.

Suggested technologies:

AngularJS (1.6, component-oriented approach would be a plus)
Angular Material
Webpack

Feel free to use starter kits, e.g. this one

To serve data, you can use either an existing backend, a mocking framework, or simply return data from services (just make sure you have services).