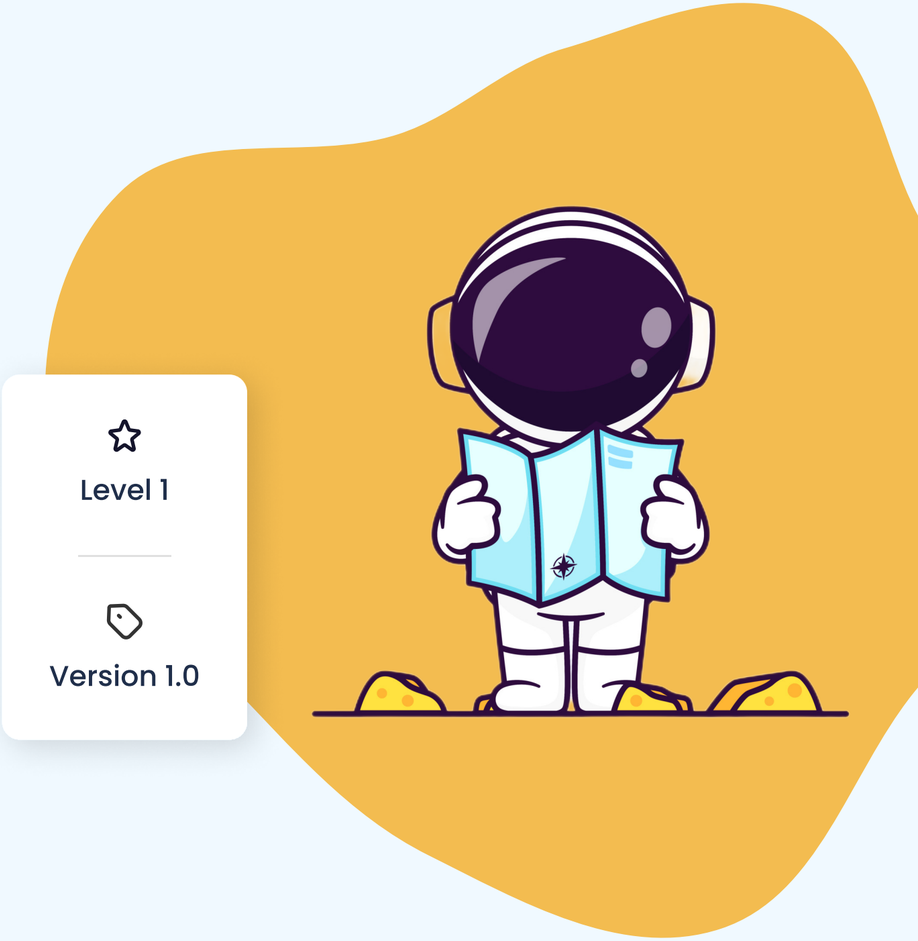
Powered by **Victor Jeman**



REACT **ROADMAP**

Created on 10 Feb 2022
Updated on 3 Sep 2022

Table of contents

01
WHY, HOW, WHAT

I believe focus and consistency is the essential skill that you need when learning something new. I also believe in going straight into the action when learning something new instead of watching countless tutorials, thus going into tutorial hell.

This document **is for you** if you want an **organized roadmap to** guide you in **learning React on your own.**

02
PREREQUISITES

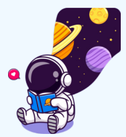
Let's talk a little more about your why. Also, let's ensure you have the tools to stay **consistent** through this journey. Let's not rely on motivation alone but rather create the necessary steps which will keep you **focused** and **engaged**.

You'll keep your code on GitHub and create a **board** where you will put all the **tasks** you need to complete for this roadmap.

03
MAIN TUTORIALS

Let's ensure you have the basic react knowledge needed to build the demo application. You'll receive some free tutorials that you need to follow before going further into the roadmap.

To **avoid tutorial hell**, follow first only the given tutorials. Once you encounter a problem you don't know how to solve, **search** for **a specific tutorial** on youtube without being tempted to watch multiple unrelated ones.



04
QUESTIONS

Answer some questions about React and Javascript to test your newly acquired knowledge.

Take the question seriously and write your answers down in a notebook. Review them weekly and try to **improve** your response with **more details**.

Table of contents

05
ACTOKEDAVRA APP

You will build a small application that will allow you to manage a list of actors. You'll be able to **add**, **edit** or **remove** actors and **sort** them based on specific filters.

The application is small and straightforward, so you will focus on cementing the React fundamentals and **best practices** instead of doing repetitive work.

Though the app is simple, you'll learn how to continuously improve it so that the structure of your app would resemble a **production-ready application**.

06
COMPONENTS

You'll learn how to split an application into small reusable components. Nowadays, all the web applications you see on the internet are sure to be built with some components-first methodology.

It's easier for our brains to handle small things, so you will learn how to identify them and build them.

07
FEATURES

A real-life web application comprises multiple features, and a feature is composed of various functionalities under the same roof. Sometimes it can be challenging to identify a feature, especially when the app you are about to build is complex.

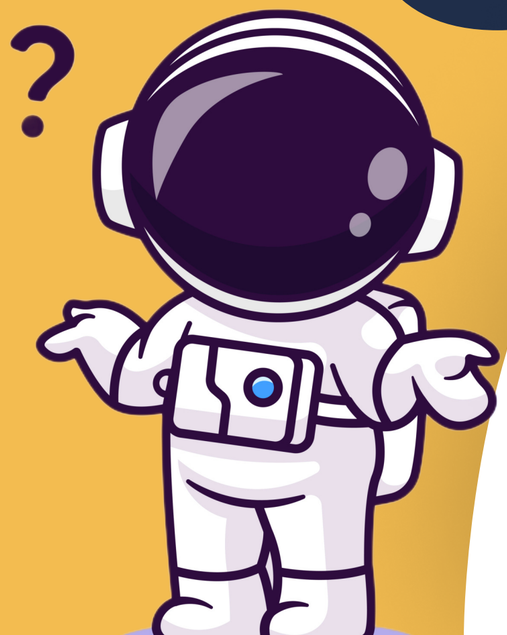
Our application has a single significant feature, and we can call it "actors." The actors' feature has functionalities like creating, editing, deleting, sorting, and listing.

08
RESPONSIVNESS

We're all talking about how well your application is displayed on different devices. You'll write specific CSS to target medium and large devices, as small devices will be covered when you create your initial components.

You'll learn how to write media queries with sass and how to organize your queries.

01



WHY, WHAT, HOW

You'll understand what this document is, what value it will bring you and how you should use it to make the best of it.

It's critical to understand why you are doing what you are doing. Before throwing yourself into any new journey, clearly understand your motivation first.

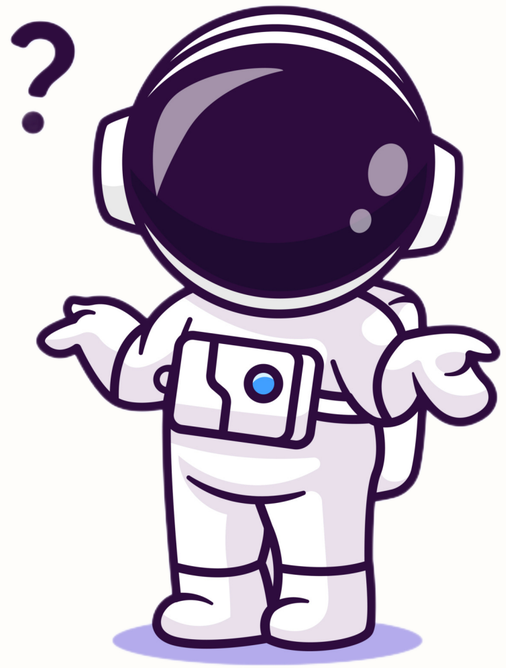
Why should I use this?

Why should I use this document? You may ask, and that's a great question. "Why" should always be on your mind when you decide to do something new.

> **TIP** Just because it's something new, it doesn't mean you need it, this pdf included. **Analyze your why carefully!**

I believe focus and consistency is the essential skill that you need when learning something new. I also believe in going straight into the action when learning something new instead of watching countless tutorials, thus going into tutorial hell.

This document **is for you** if you want an **organized roadmap to** guide you in **learning React on your own**.

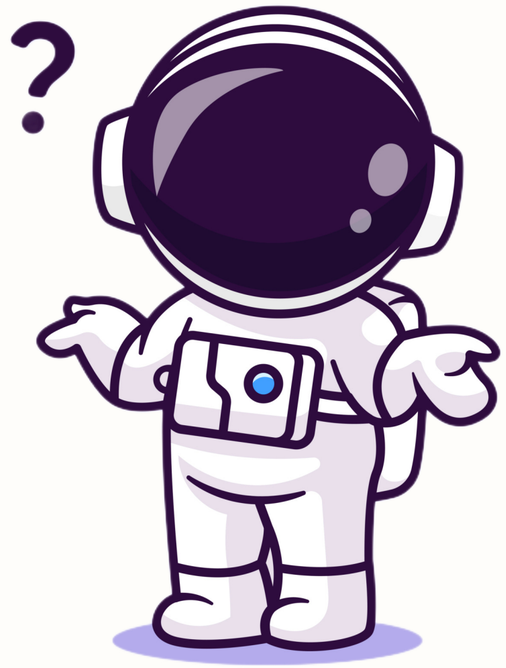


Important

This is not a step-by-step tutorial(at least not yet) that will guide you in every detail. You have room for exploration and thus room for error, but that's how you're going to learn in real life.

How to use it?

Use this document as a roadmap to guide you in a specific direction; still, you must be ready to search for extra tutorials if needed. We are at version 1. 0 of the roadmap and don't yet have custom video tutorials for each section. Stay tuned for version two.



PREREQUISITES

Before starting your journey let’s make sure you have all the needed tools.

02

Why do you want this?

If you are ready to strengthen your React knowledge, **it's critical to define why you do what you do**.

Why do you want to be better at React?

- Do you want to switch technologies because you are tired of what you are using now?
- Do you hope to get a raise if you'll prove to your employer that you know another framework?
- Maybe you are a student, and you want to land a job?

I can't stress enough how **important** it is for you to **clearly define why you want to proceed with learning React**. The why should drive you daily.

Your **motivation will fade**, and the **excuses will pile up very fast unless you have a damn good answer to your why question**.

So, put a cup of tea and sit down some time in silence and think very hard about why you want this.

> **TIP** When you encounter days when you don't feel like spending time on this application, remember the WHY and do it anyway.

Make a schedule

The second critical point for you is to make a schedule. Don't drift mindlessly through this document. If you're in, then let's do this right.

Decide **how much time you can offer** for this project; is it 30, 45, or 60 minutes a day? Use this simple phrase to define your schedule

I will **[behavior]** at **[time]** in **[location]** for **[duration]**.

For example, I will **learn React.js** at **9:20 am** in **the office** for **45 minutes** right after I drink my coffee.

TIP Combine this new process you are trying to establish with an existing habit that you rarely miss, like drinking coffee.

TIP Put this new habit in the calendar and block that specific slot so that there is a smaller chance people will bother you.

TIP Close all the unrelated tabs, silence your phone, and focus all your attention on this project.

Github

Make sure you have a GitHub account and know the git basics.

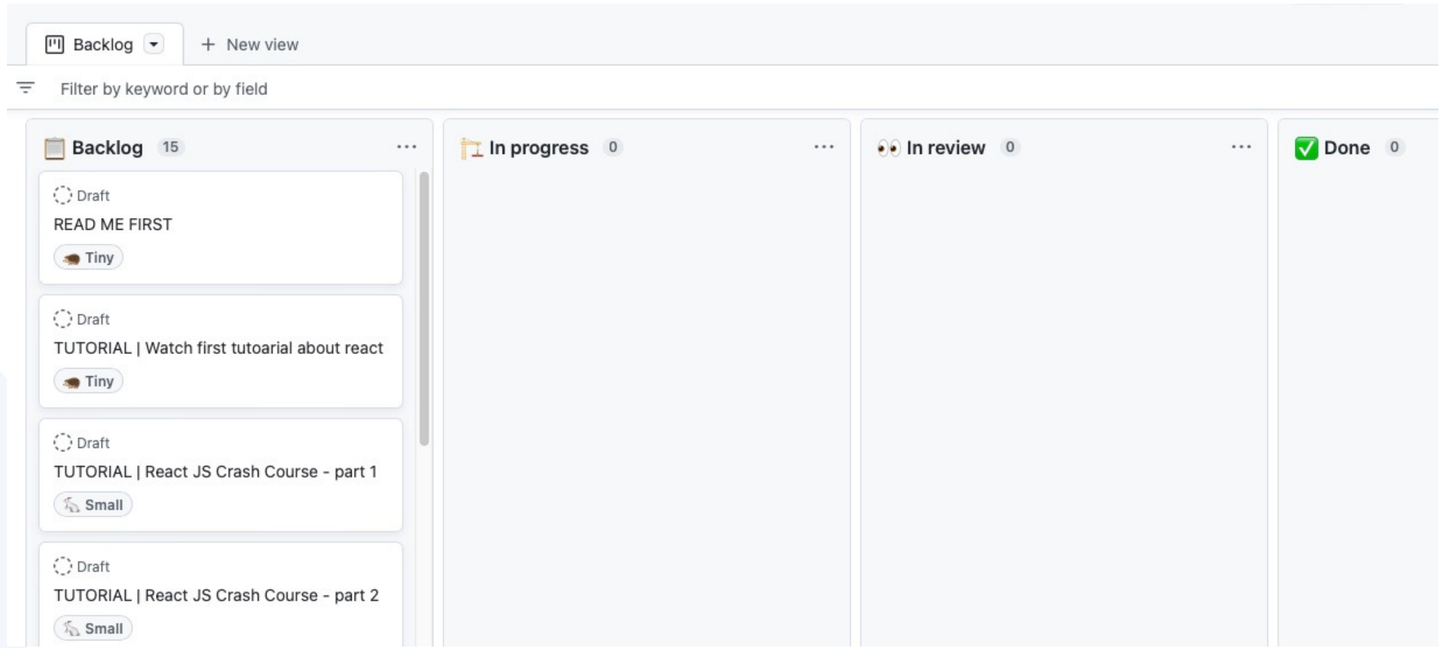
GitHub Signup

VIDEO | Git, GitHub, & GitHub Desktop for beginners

Be organized

Create a board and the tasks based on how much time you can allocate daily for this project.

GitHub board example



Be organized

The basic GitHub board example I've created will show how you could split the work from this roadmap into small tasks.

My assumption is that I have 40 minutes daily to work on ActoKedavra app, so I created the tasks accordingly.

> TIP You should have **a daily dose of "I've got this"** by dragging a task from "todo" to "done".

The reality's such that inevitably you'll have **a task** that **takes more than 40 minutes**. If you don't know in advance how to split them into more manageable pieces that take less than 40 minutes, it's ok.

Drag the task into the "in progress" column, start working on it and once 40 minutes have passed, create another task with the same name but with "- part x", at the end, e.g. ***"Implement buttons - part 1"***, ***"Implement the modal - part 2"*** and drag that task into the "done" column while keeping the main task in progress.

Be organized

Leave **a couple of words** in the task that you just dragged into the "done" column **about what you have managed to accomplish in these 40 minutes**, then you are done for the day.

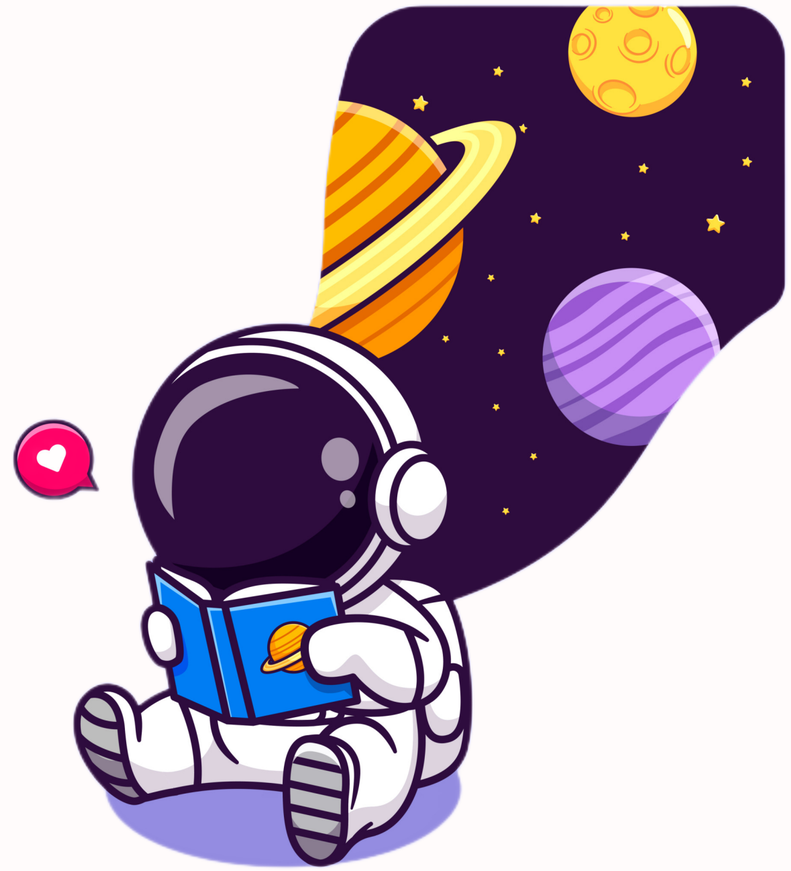
> TIP Being **consistent** in small doses **is more effective** than being very **productive from time to time**.

Don't feel like working today; it's one of those days? It happens to the best of us. On days like these, chose an article about React that takes 5-10 minutes to read. Create a task for that article, read the article, leave the link in the task you just created and move the task into done.

> TIP Do something, **don't break the** "I've got this" **chain**.

> TIP Don't like how the GitHub board looks? Use anything you want, and it doesn't matter what you use. **It matters to stay organized** with a well-defined to-do list.

03



MAIN TUTORIALS

Let's ensure you have the basic react knowledge needed to build the demo application. You'll receive some free tutorials that you need to follow before going further into the roadmap.

To avoid tutorial hell, follow first only the given tutorials. Once you encounter a problem you don't know how to solve, search for a specific tutorial on youtube without being tempted to watch multiple unrelated ones.

About main tutorials

Watch the main tutorials carefully and write the code for all practical examples in a GitHub Repo.

There are secondary tutorials throughout the document to help you solve more specific use cases.

When you see a 📘, it's time to learn something new and exciting.

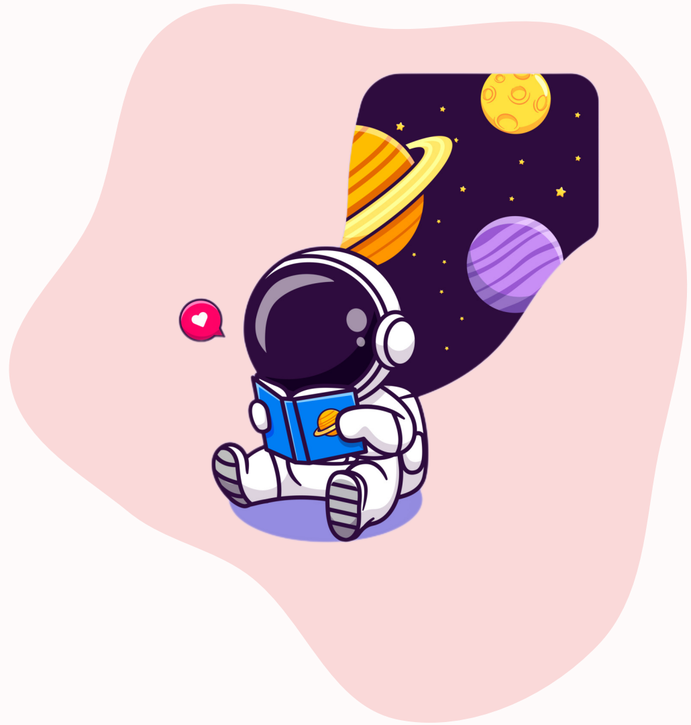
> **TIP** Don't forget to create tasks for watching the tutorials. **Everything** you do regarding this roadmap **should be a task**.

About main tutorials

Watch the main tutorials carefully and write the code for all practical examples in a GitHub Repo.

There are secondary tutorials throughout the document to help you solve more specific use cases.

- Video | React in 100 seconds
- Video | Learn React JS in just 5 MINUTES
- Video | Learn JS Crash Course



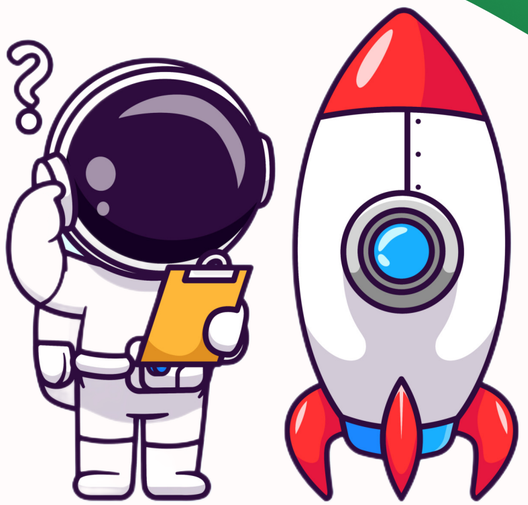
03

QUESTIONS

- "I know how to code, but I'm not good at explaining things."
- "I understand it, but I can't explain it."
- "I can do it, but I don't know why I'm doing it this way."

Don't grow into a developer that uses phrases like these. Only when trying to explain something to someone else will you see if you truly master your skills.

I encourage you to explain all the questions from this section as clearly as possible.



DO

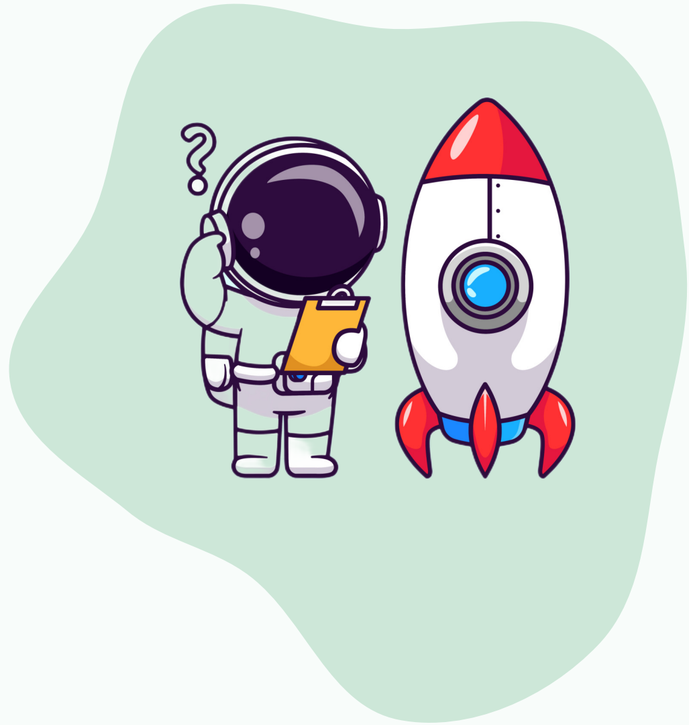
- Copy the question to a different document and answer there.
- Review them weekly and improve the answers.

DON'T

- Skip the questions.
- Take them lightly.
- Avoid the level 3 challenge.

About questions

After watching the main tutorials, answer the questions using your own words.
For some questions, imagine you are explaining it to



Level 1

easy - someone with web development experience

Level 2

meidum - someone with minimal web development experience, e.g., a student

Level 3

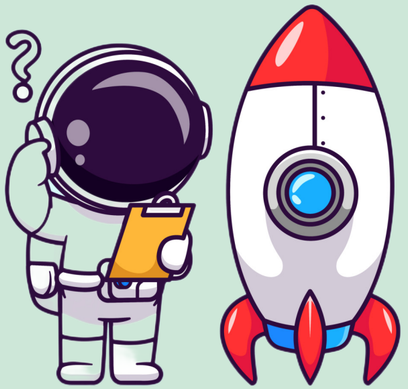
hard - someone with zero web development experience, e.g., a history student

About questions

TIP Explain the basic notions about a computer and other components the listener may need to understand before explaining the main question.

TIP When necessary, split the question into multiple parts and explain them independently before combining them and trying to explain the main question.

TIP Explain the question using real-life examples that the listener may know.



03 QUESTIONS

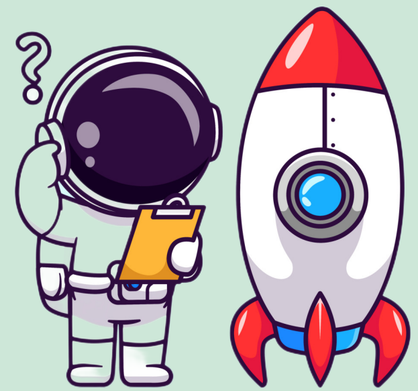
Q What is React.js? Level: 1, 2, 3

Q What is the difference between imperative and declarative programming? Level: 1, 2, 3

Q React is declarative or imperative, and why? Level: 1

Q Can I build full-stack applications with React, and how? Level: 1, 2

Q What is JSX, and what does it allow you to do? Level: 1, 2



03 QUESTIONS

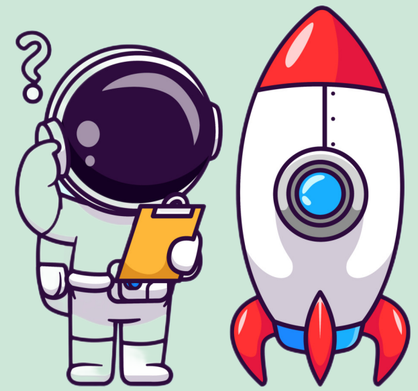
Q What is a component? Level: 1, 2, 3

Q What is a presentational component? Level: 1, 2

Q What is useState, and for what can I use it? Level 1

Q How can you style things in React? Level: 1

Q What do mutable and immutable mean in javascript? Level: 1, 2, 3

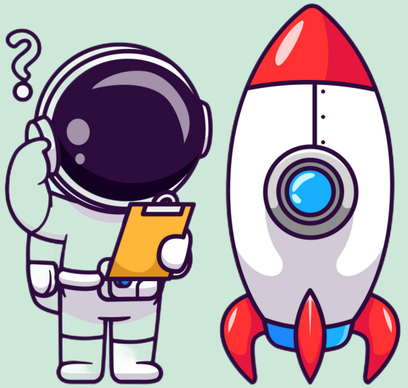


03 QUESTIONS

Q When using useState hook the created state is mutable or immutable? Level: 1

Q What is a controlled component? Level: 1

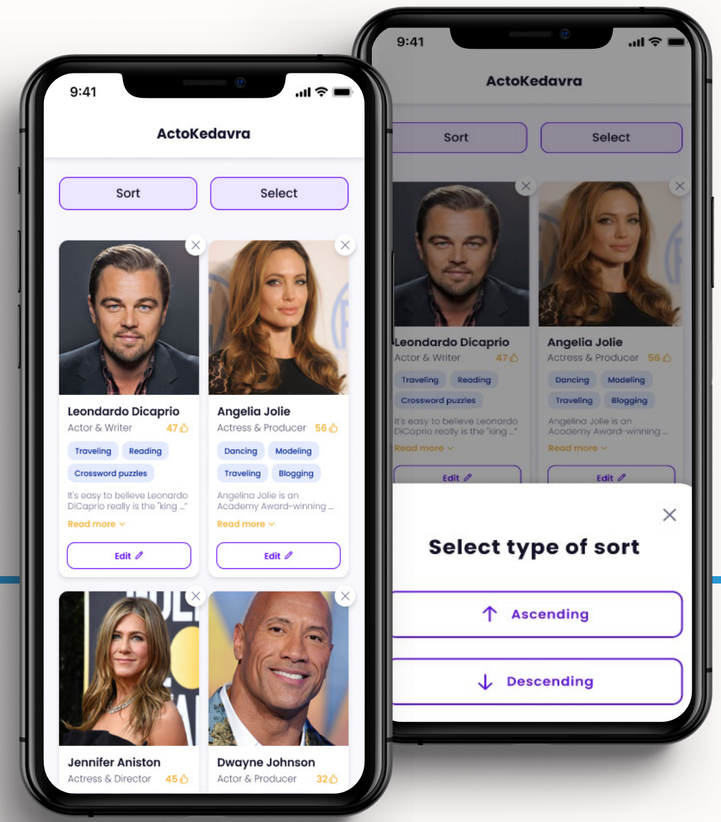
Q Is it worth learning React and why? Level: 1, 2, 3



ACTOKEDAVRA APP

Do things to your favorite actors.

04



What will you build?

You will build a small application that will allow you to manage a list of actors. You'll be able to **add**, **edit** or **remove** actors and **sort** them based on specific filters.

The application is small and straightforward, so you will focus on cementing the React fundamentals and **best practices** instead of doing repetitive work.

Though the app is simple, you'll learn how to continuously improve it so that the structure of your app would resemble a **production-ready application**.

UI COMPONENTS

You'll learn how to split an application into small reusable components. Nowadays, most web applications you see on the internet are sure to be built with a components-first methodology.

It's easier for our brains to handle small things, so you will learn how to identify and build components.

05

Intro

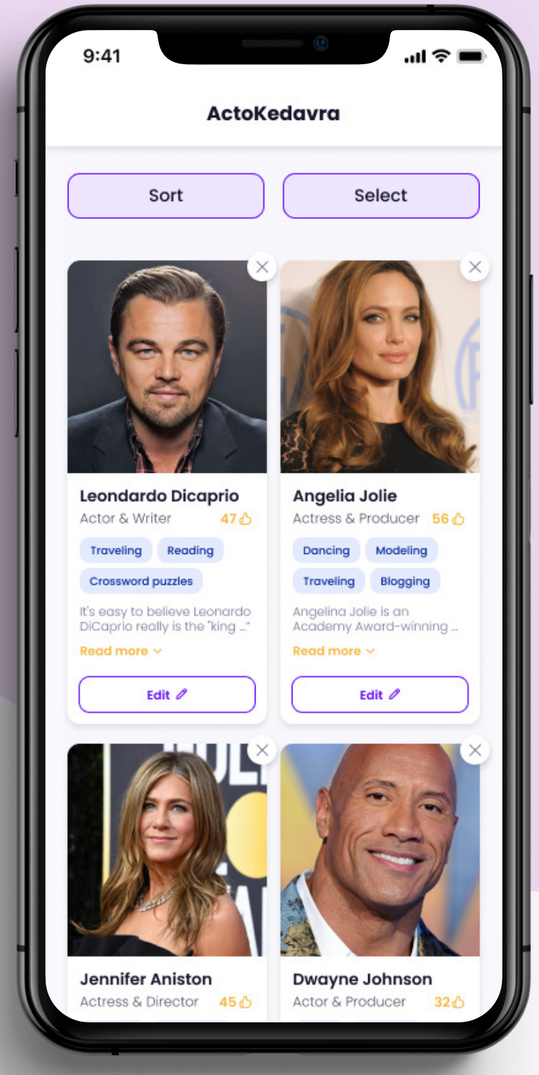
Before working on functionalities like creating, deleting, or editing actor s, you'll focus on completing most of the needed presentational components.

The presentational components are stateless functional components that are only concerned with rendering data to the view

You are about to build several components that won't connect directly to the store but instead wait for other components to send the necessary data through props.

Components like:

- <PageHeader />
- <PageFooter/>
- <Button />
- <ShowMore />
- <Badge />
- <Modal />
- etc.



Intro

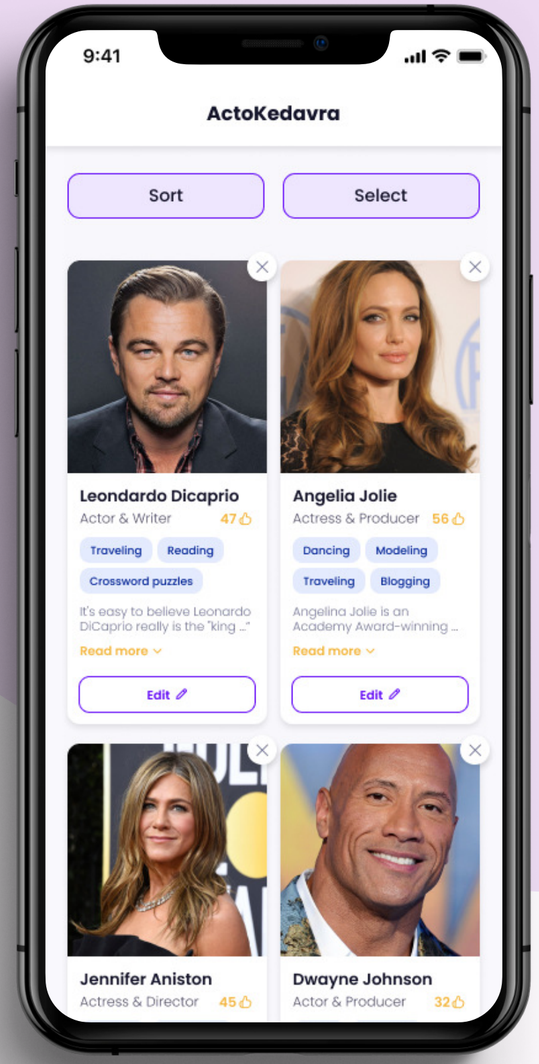
There are dedicated tools for building UI components and pages in isolation; one such tool is the storybook. View the *"Storybook for frontend development"* video to understand better what the storybook is trying to solve.

You will not use storybook just yet as this may introduce more difficulties at this point. Still, you will split your app into multiple components and import all of them into the App.jsx file.

Storybook for frontend development

Focus on **small screens first**, don't worry about adjusting your components for big screens. This is an educational project, and we'll do things in a more organized way.

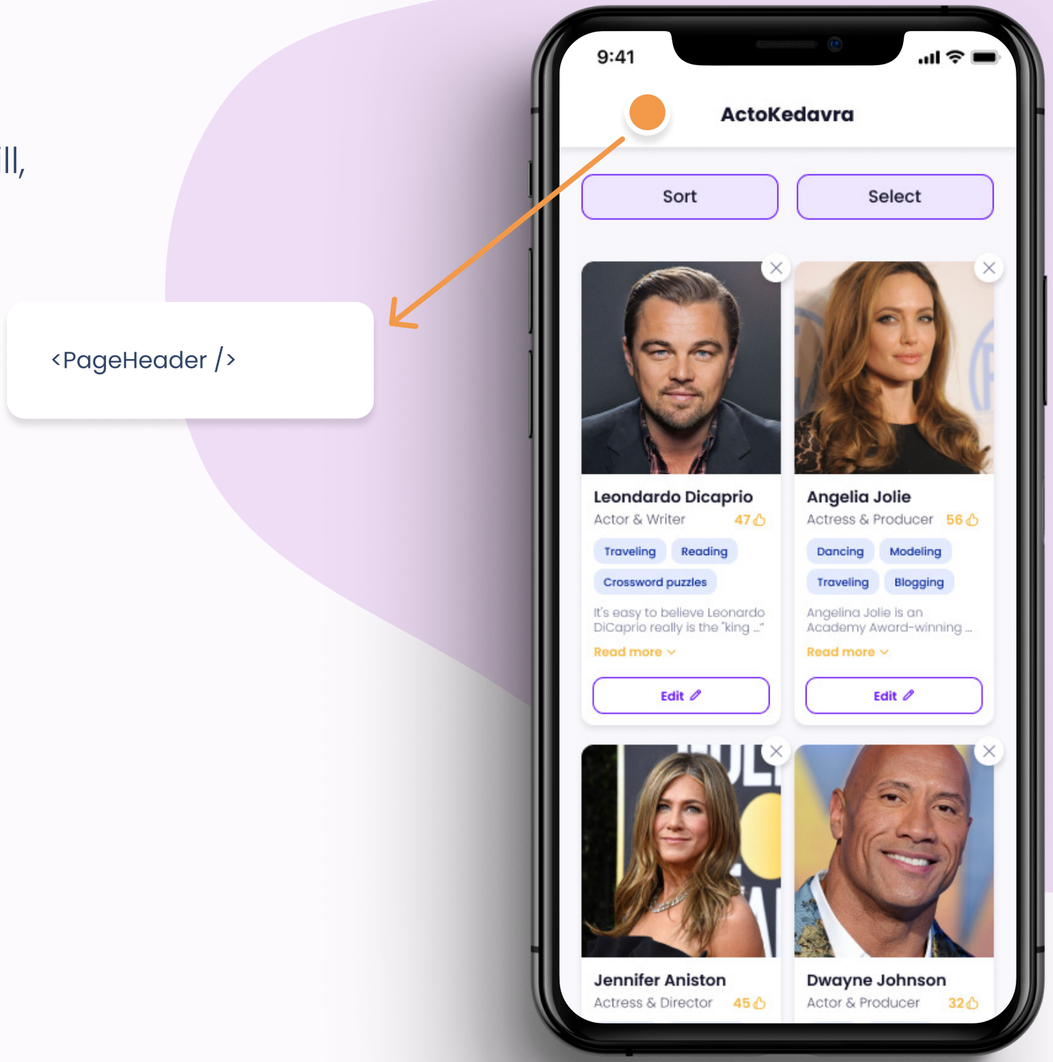
In a real-life project, it's best to handle the responsiveness of an element right away, you also won't always have the time to create all the UI components before proceeding to functionalities.



Page header

The page header is a dead simple component that displays only the application logo in the center but still, it's helpful to extract that piece of UI in a separate file.

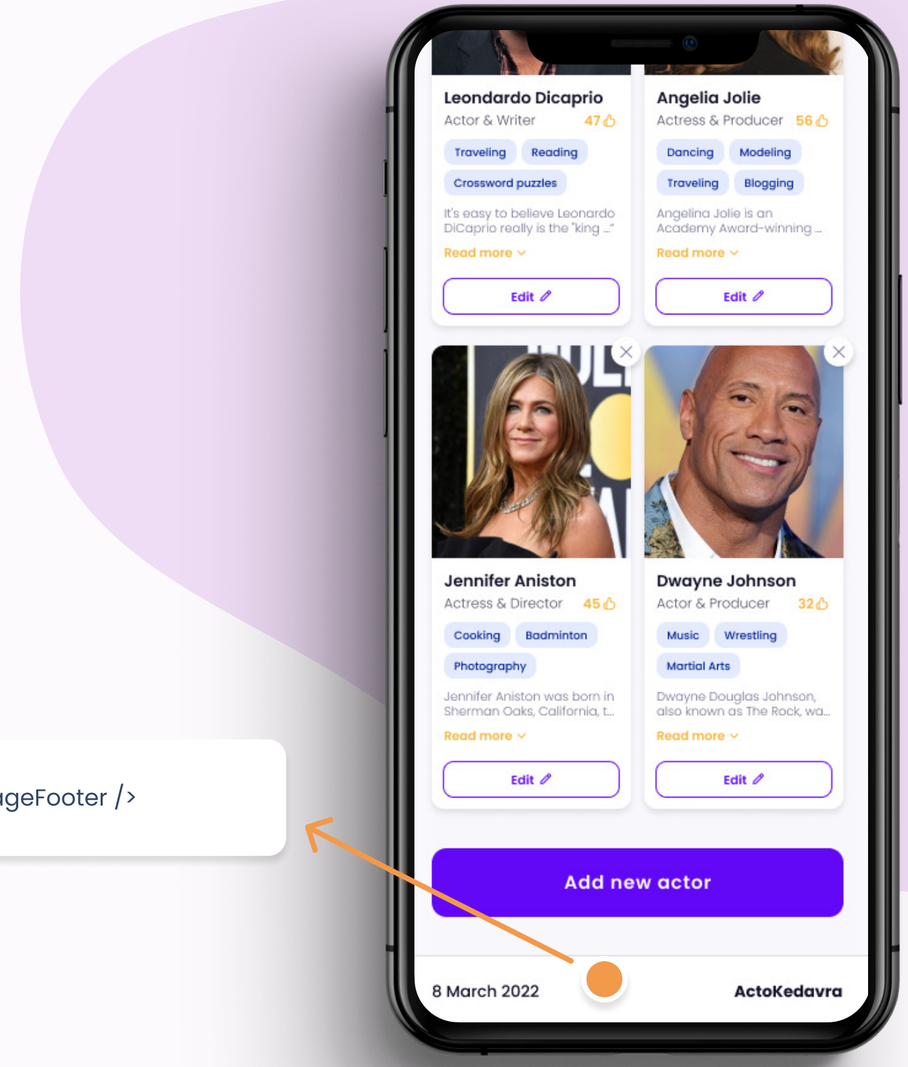
Notice also that it's fixed, meaning the header will remain visible no matter how much we scroll.



Page footer

This component is very similar to the page header in terms of complexity.

The date in the footer should always be the current date.

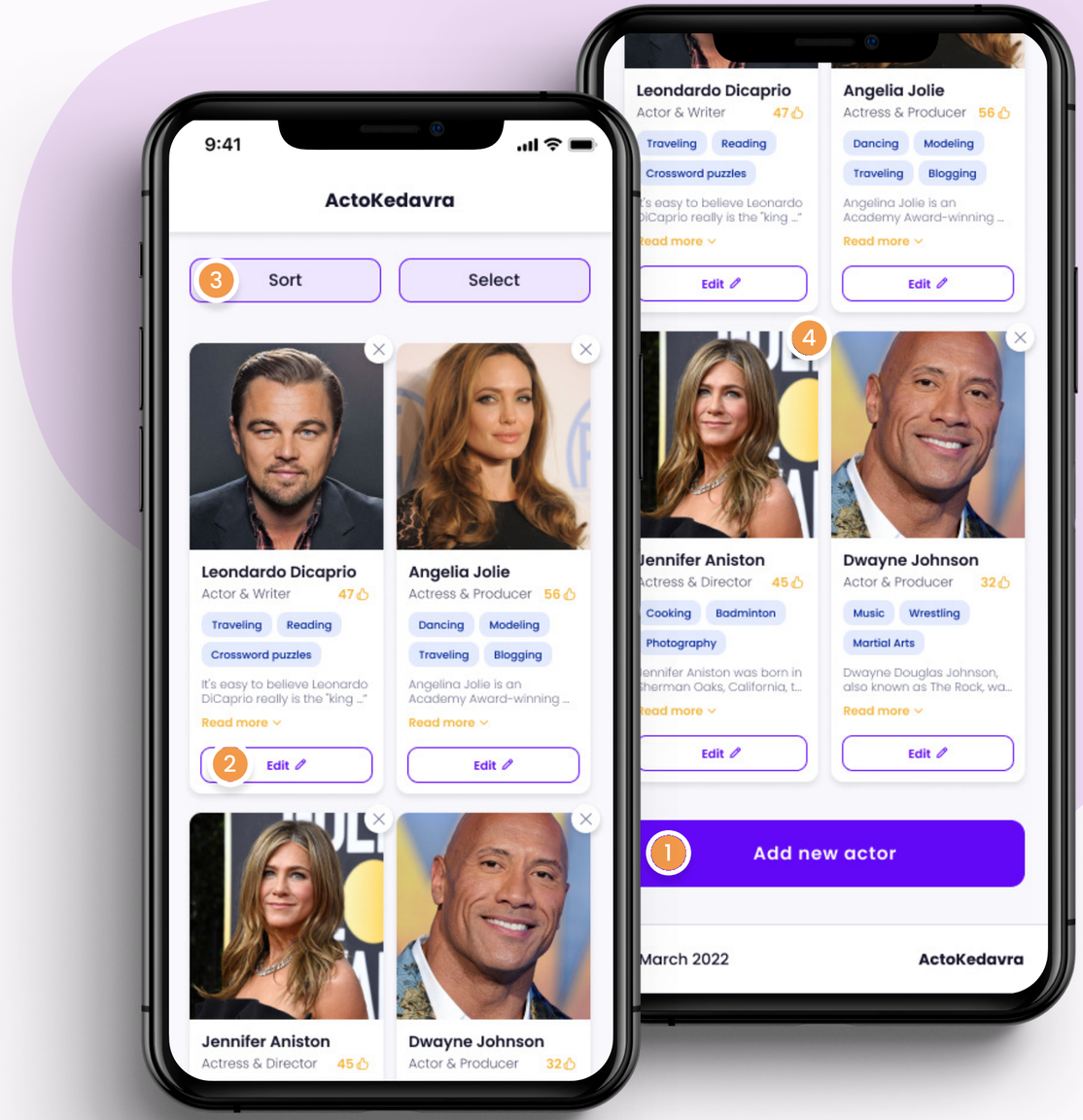


Buttons

Looking at the Figma design, you may see multiple types of buttons.

Think about how to create a single <Button /> component that could change the appearance based on properties.

- Figure out how many kinds of buttons you have in your application and add the list to your GitHub task.
- The Button component should give you all the needed UI possibilities from Figma.
- Make sure to address the button's hover, focus, active and disabled states.



Buttons

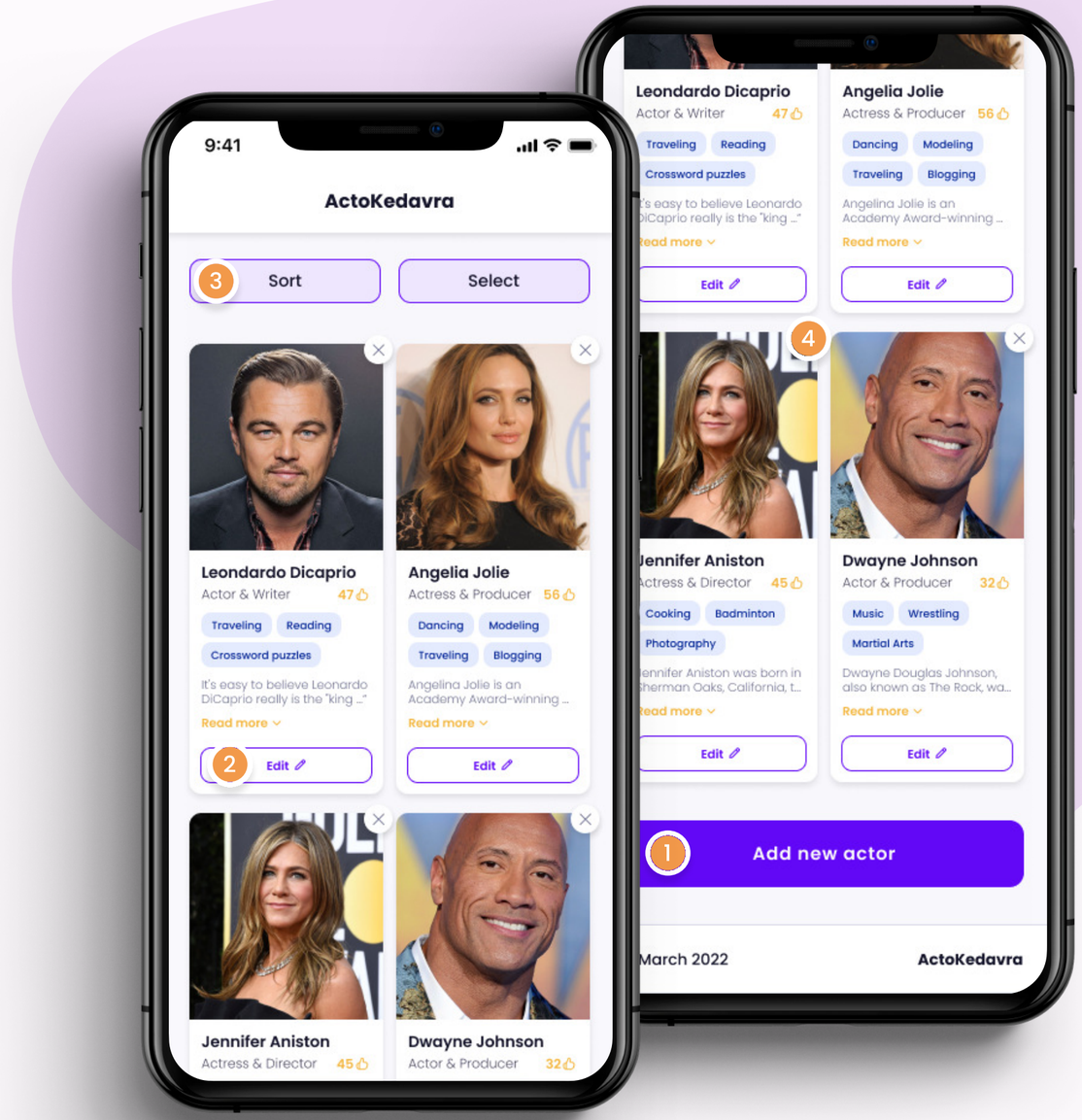
On the right, you have some suggestions on how you could approach the Button component.

1. <Button type="primary" size="large">

2. <Button type="secondary" size="medium">

3. <Button type="help" size="medium">

4. <Button type="circle" size="small">

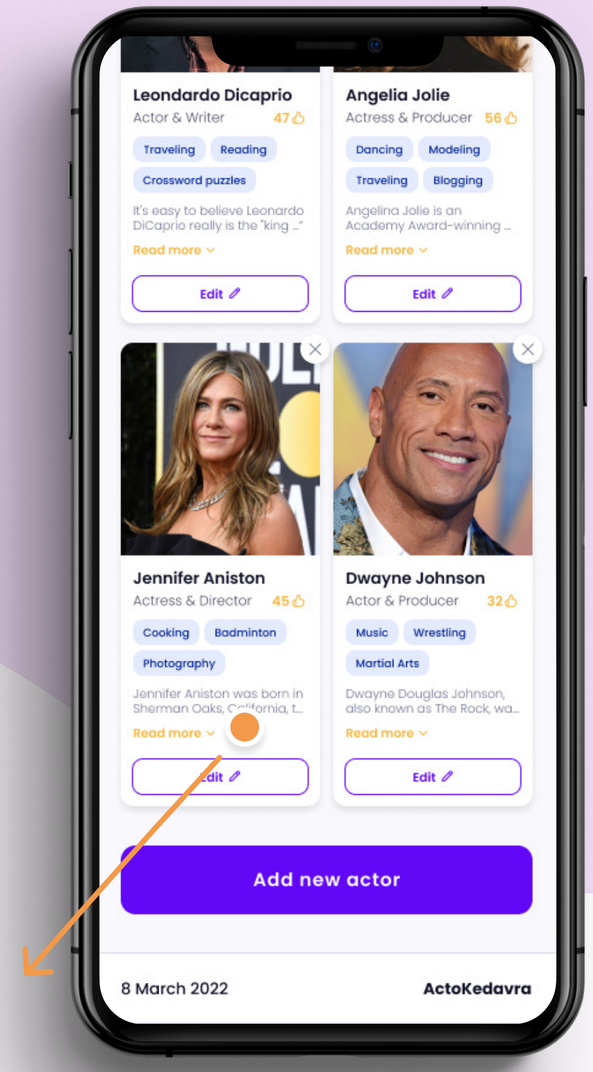


Show more

The "show more" functionality, while used in the actor thumbnail only in our design, theoretically, if the application grows, we can use it in other components that may need the same functionality of displaying less text with the ability to show more.

We can implement a generic component that will receive some text and then, based on other parameters, display a portion of the text with the ability to show it all on click.

This generic component shouldn't be heavily styled and should allow other components to style it based on their need. For example, the ActorThumbnail component will choose to style the Read more label with the color orange and font-weight bold.

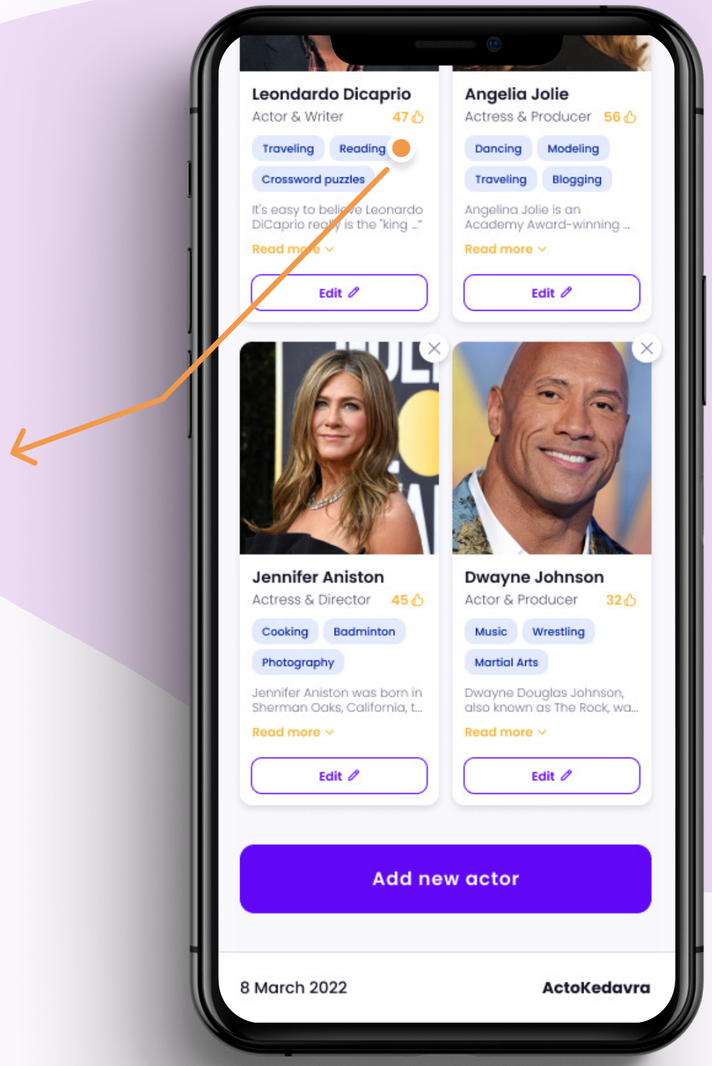



```
<ShowMore
  labelStart=”Read more”
  labelEnd=”Read less”
  limit={40}>
  It's easy to believe Leonardo DiCaprio
  really is the "king” of the movie industry.
  He’s also good looking.
</ShowMore >
```

Badges

You could leave this badge as part of the actor thumbnail component, but if you want to improve your component-splitting skill, create a separate component and make it reusable.

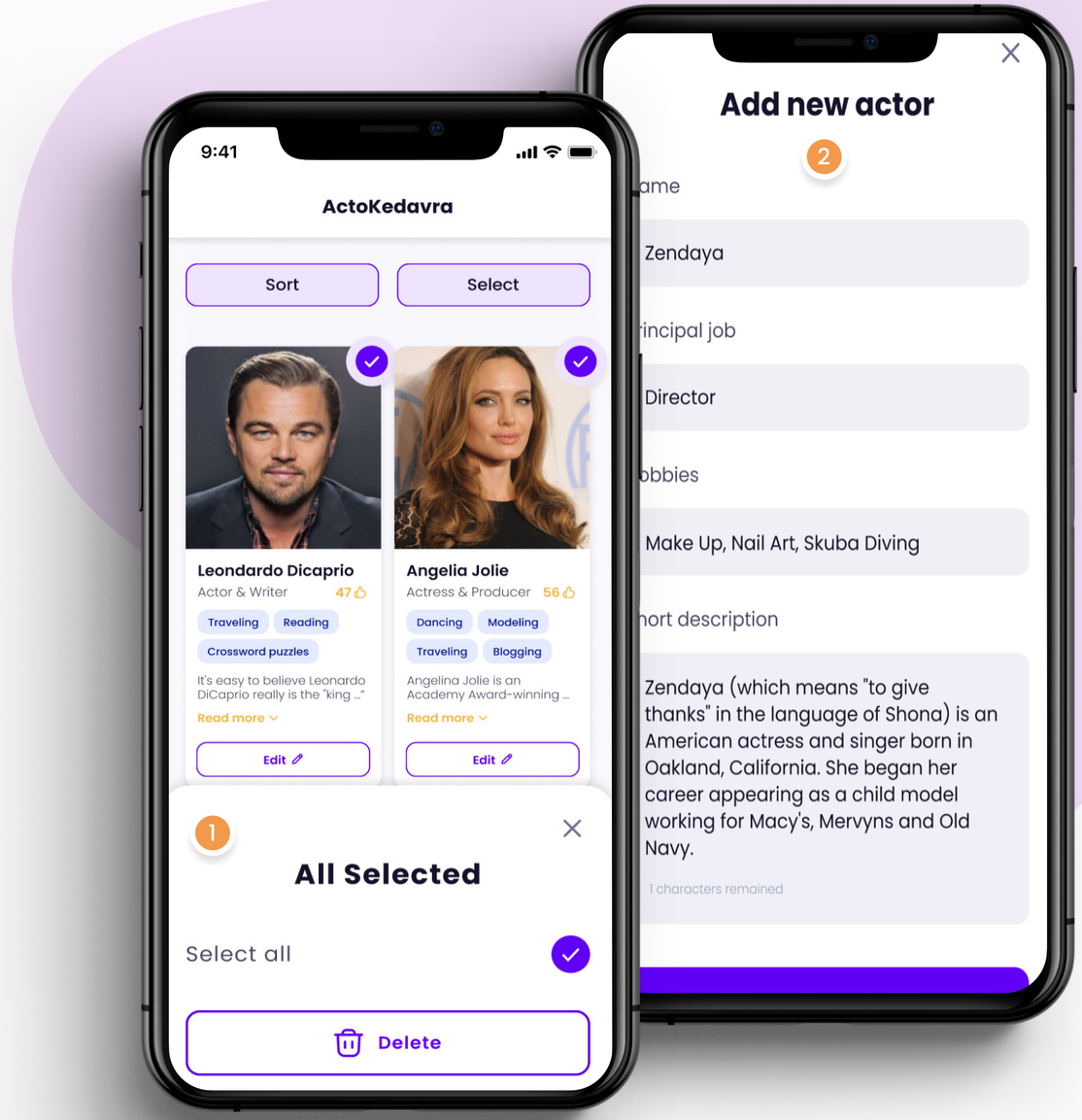
```
<Badge> Reading </Badge >

<Badges  badges={['reading', "dancing"]} />
```



Modal

This application relies on modals to display a good portion of information, so you should create a generic Modal component that you can reuse.

- Create a modal component that can take different types of children.
- The modal should have a close button as part of its functio nality.
- The close button should be optional, meaning you can hide it if you don't need it.
- Close the modal when you click outside of it.



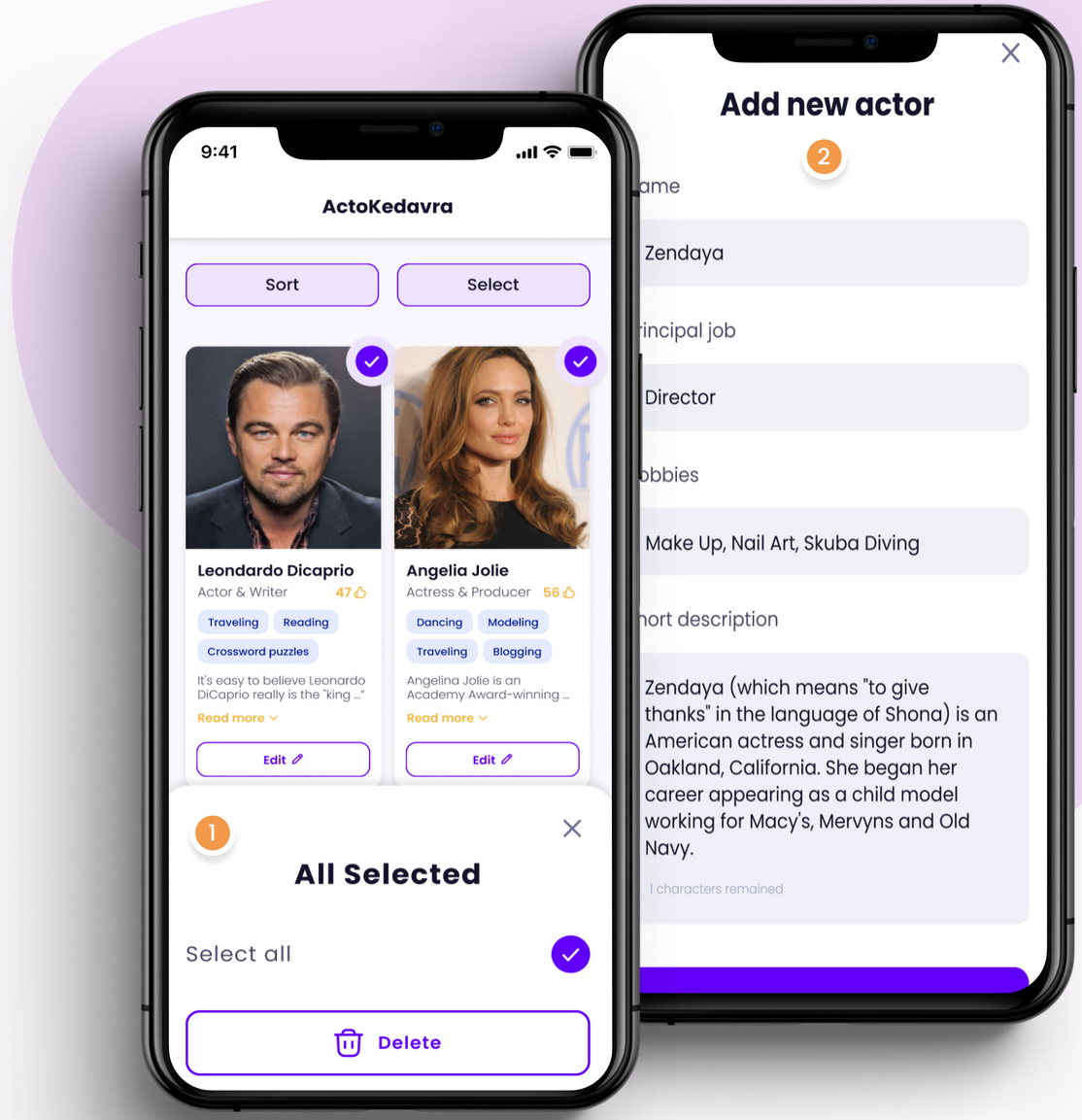
Modal

Notice that the modal should accommodate the content inside of it.

1
```
<Modal title="All Selected">
  // Select all functionality inside
</Modal>
```

2
```
<Modal title="Add new actor">
  // Add new actor functionality
</Modal>
```

2
```
<Modal title="Edit actor">
  // Edit existing actor functionality inside
</Modal>
```

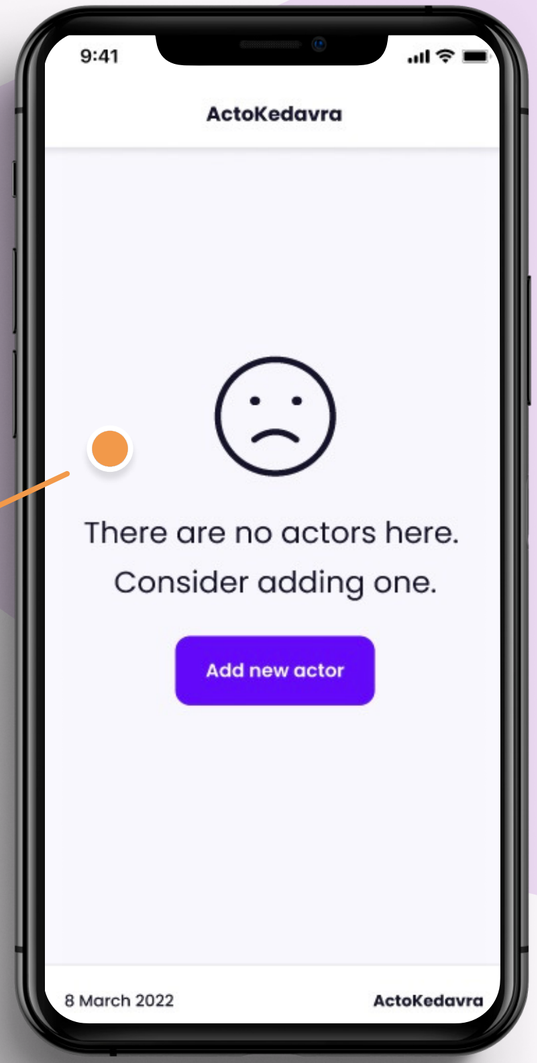


No actors

Create a component that will be displayed when no more actors are on the list.

Remember that we have a generic component for the buttons, so you'll need to reuse the <Buttton /> here.

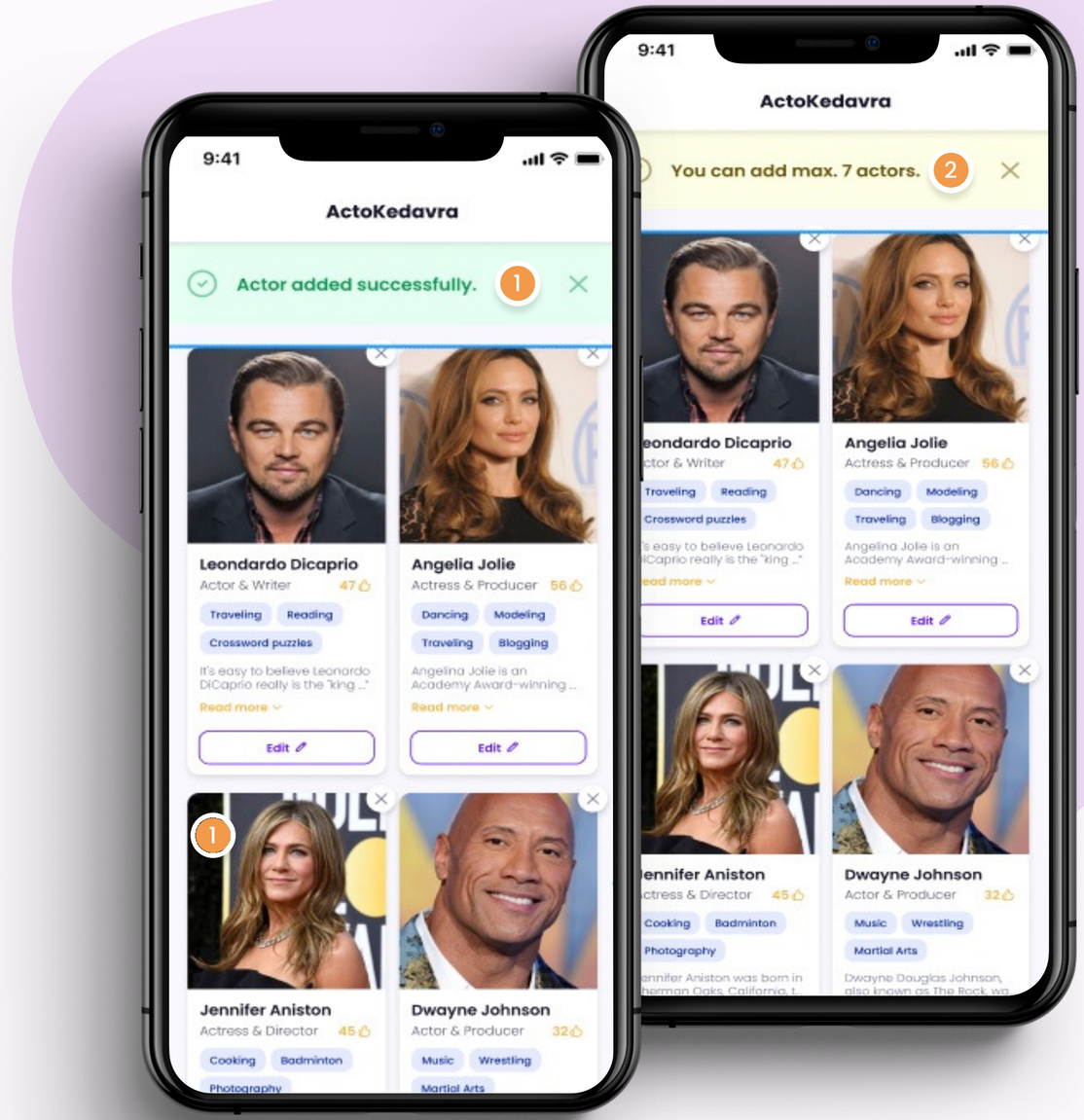
<NoActors />



Alerts

The alert is a more simple version of a modal. It can have only a message and an icon in our case.

- Show a close button that will close the alert.
- Add the ability to change the alert's appearance to have three types: success, warning, and error.



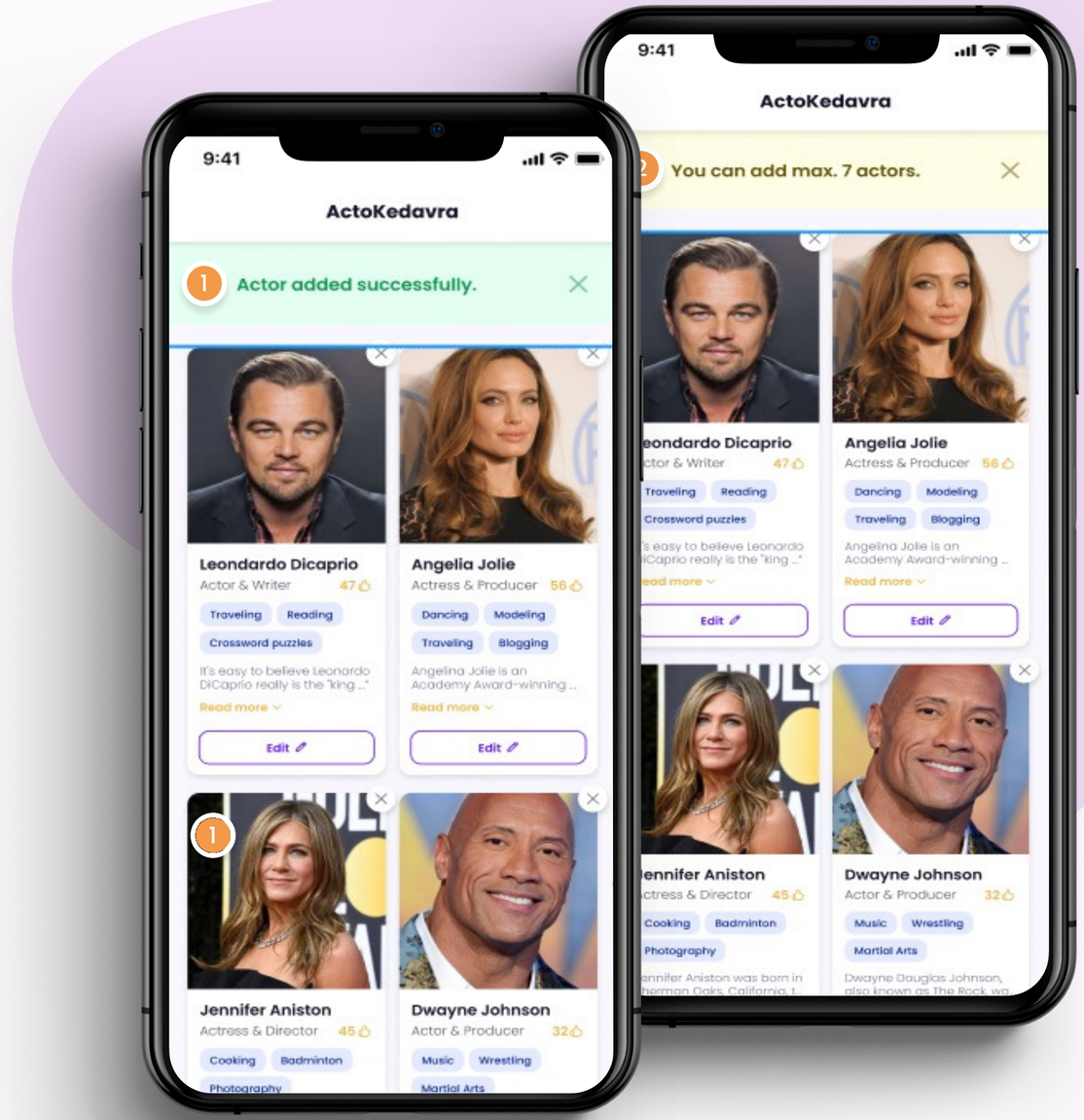
Alerts

See some component examples below.

1. <Alert type="success" autoclose={true}>
   Actor added successfully
</Modal>

2. <Alert type="warning" autoclose={false}>
   You can add max. 7 actors
</Alert>

3. <Alert type="error" closable={false}>
   You're changes were not save
</Alert>

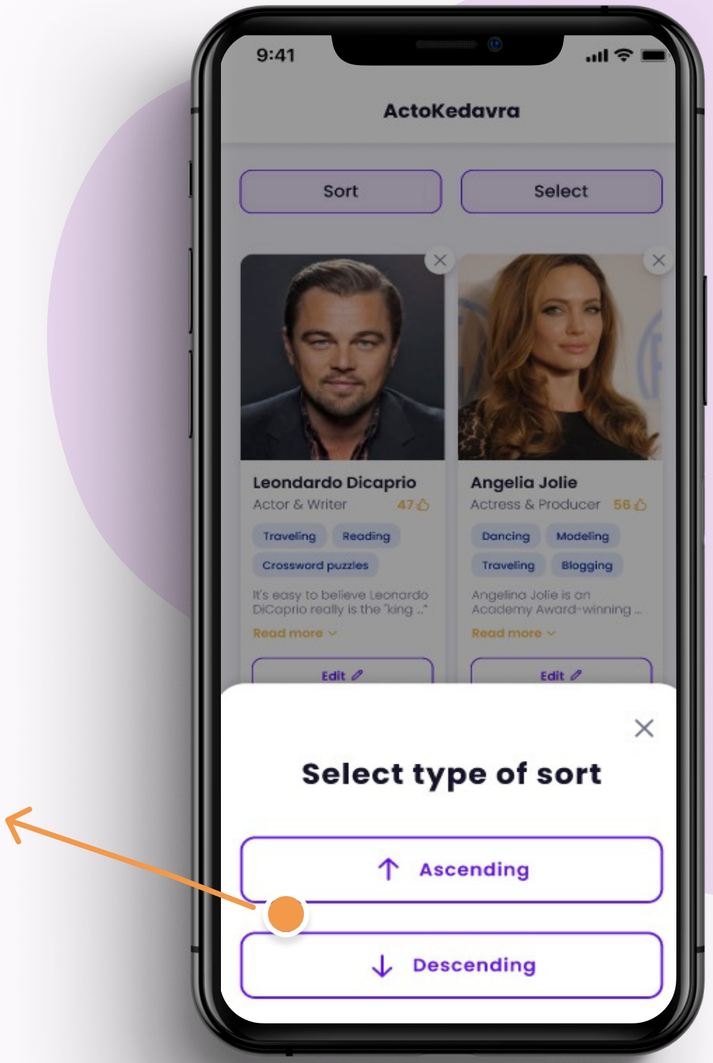


Actor sort

Implement the soring component. It doesn't have to sort any content at the moment but think about what methods must work so that it would connect well with the list of actors. This component will be inside a modal, so you don't need to worry about the background or borders or the tile.

Remember that you already have the buttons implemented at this point, so this component is mostly about gluing together other smaller components.

<ActorSort />

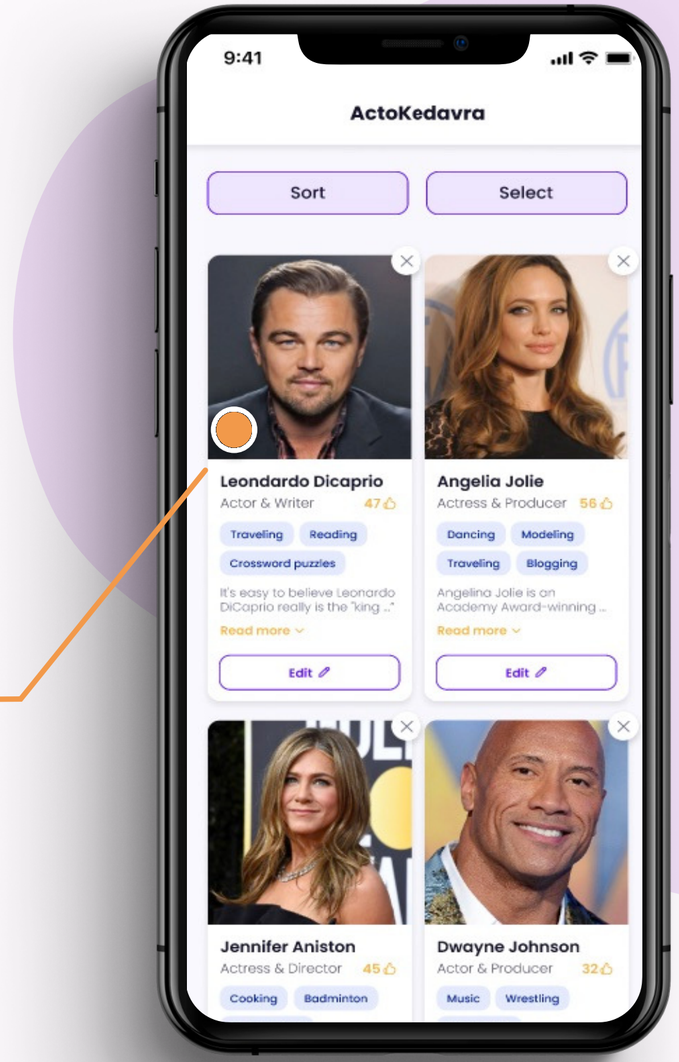


Actor thumbnail

The actor thumbnail component can be created by reusing the <Button />, <Badge/>, and <ShowMore /> components.

You'll have to use some new elements for the image, title, subtitle, and score, but at least you are not starting from scratch.

<ActorThumbnail actor={actor} />



Strech your knoledge

Learn about controlled and uncontrolled components in React.

Understand the differences between these approaches and think of examples when to use them.

[Controlled vs. uncontrolled components in React]

[React Forms Explained]

[How to make a Basic Form using React Hooks]

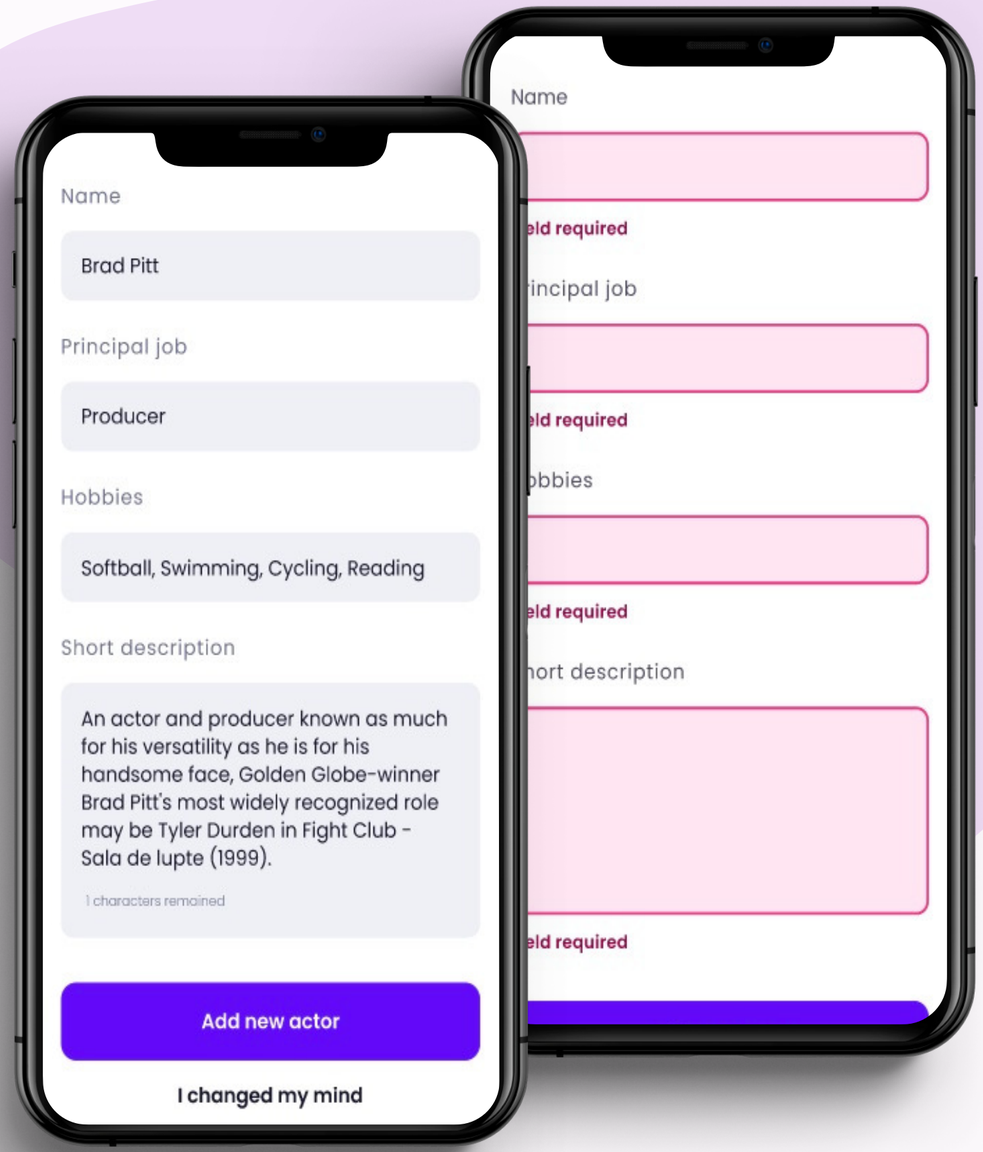
Q

When one approach is better than the other?

Actor form

You need a form that will add new actors. You could create a more generic input component, but for now, I think it's ok if you make the entire form as a single component. Don't forget that the form is inside a modal, so you don't need to worry about the background and borders.

- Create the add new actor form.
- Don't add any actors yet but implement all the required handlers.
- The short description field should accept only 180 characters.
- Disable the input and don't allow the user to type anymore if there are more than 180 characters.
- The 180 number should decrease every time the user hits a key.
- All fields are required.
- When you hit the submit button without adding any data inside the fields, display a red border, and a message



FEATURES

Now that you have most of the presentational components implemented, it’s time to glue them together so that you would have some functionalities.

The term "feature" in front-end development can be confusing. In this section, I've referred to features as a handful of functionalities you can do around actors.

You'll see later when discussing react project structure that we use the same "feature" term when structuring the app, but it represents something more significant than a simple functionality.

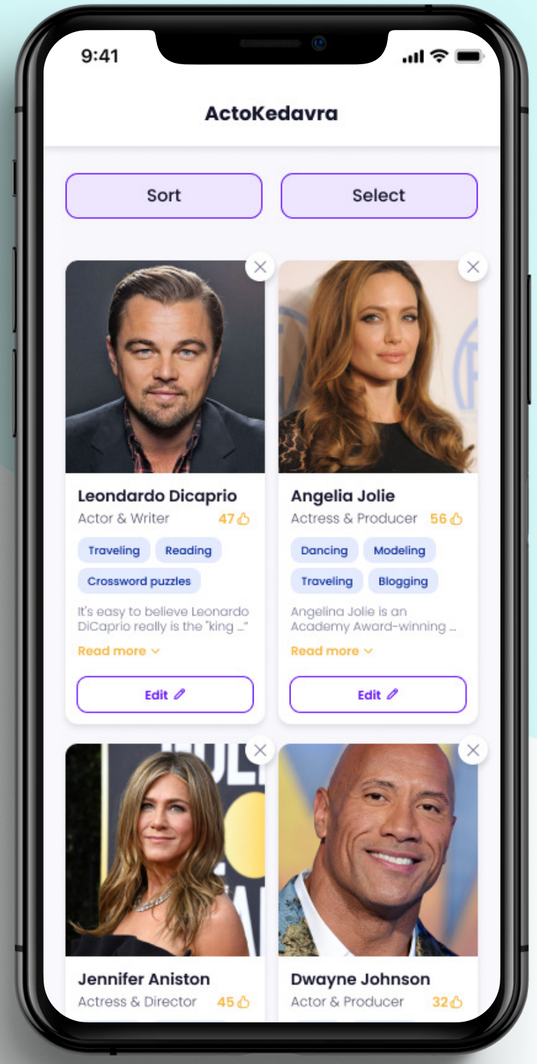
But let's not get ahead of ourselves; for the moment, know that you are about to build some basic functionalities for actors.

06

Strech yout knowledge

Before implementing some features, learn how to create a new page so that you can move your style guide to a separate page. The following tutorial should guide how to use React Router to create an additional page.

Video | React Router V6 Tutorial

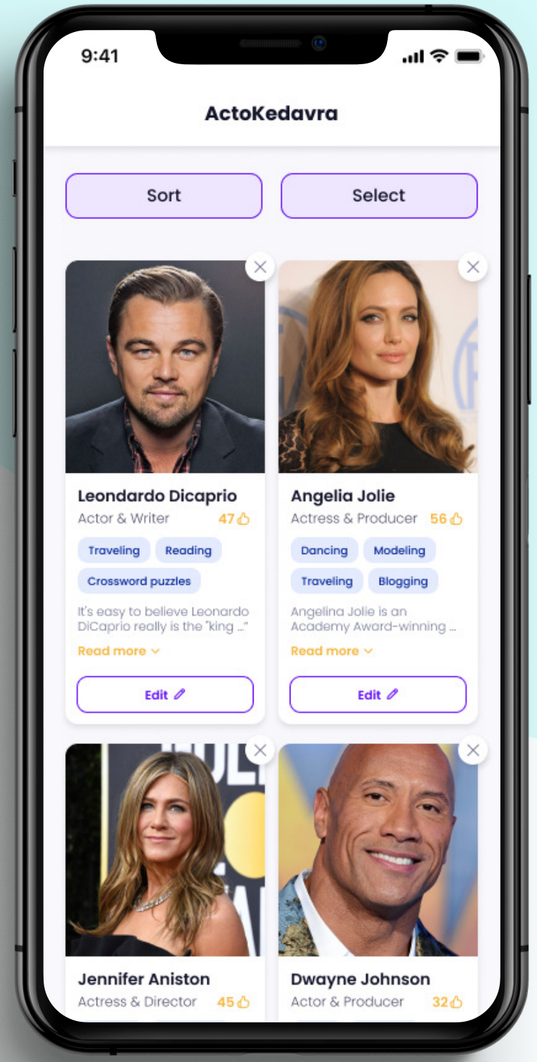


Dedicated page for your styleguide

While you created most of your small UI components, you started, in some sense, a style guide. Now that you need to implement features moving all your components from **App.jsx** into a separate page is best.

- Create a page called **StyleGuidePage**.
- Move all your components imports in there.
- Make sure that if you change the URL path to "**/styleguide**," you'll see your components.

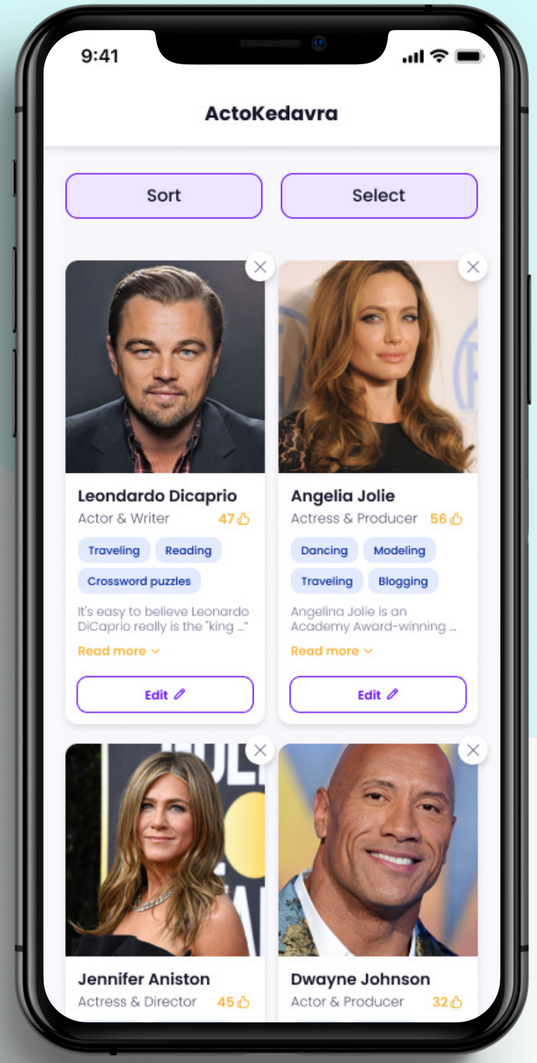
In doing this, you'll have your App.jsx file ready to be used for implementing the main functionalities of your application.



Display the actor list

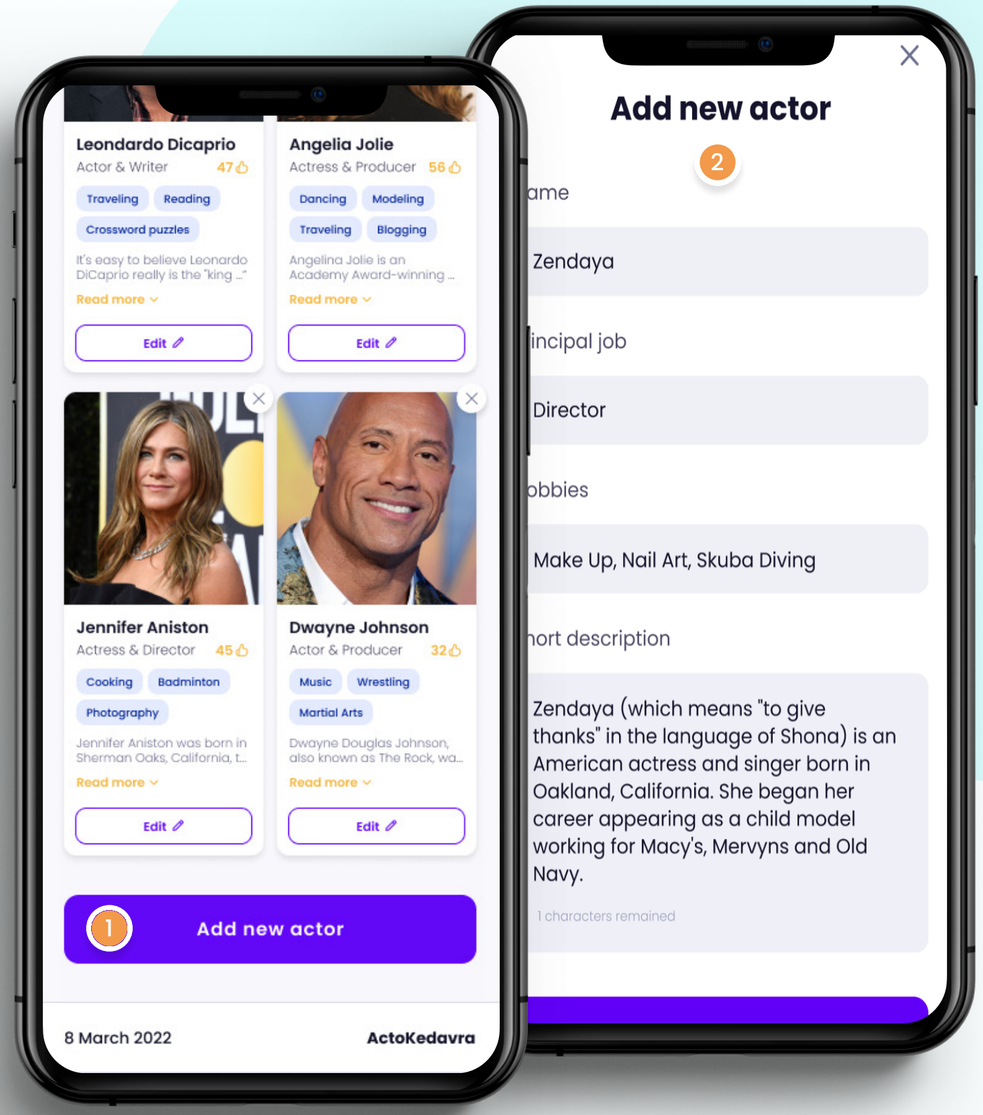
Display the actor list using a hardcoded JSON. I recommend using json-server for a fake API. Use axios for API requests.

- JSON Server
- axios documentation



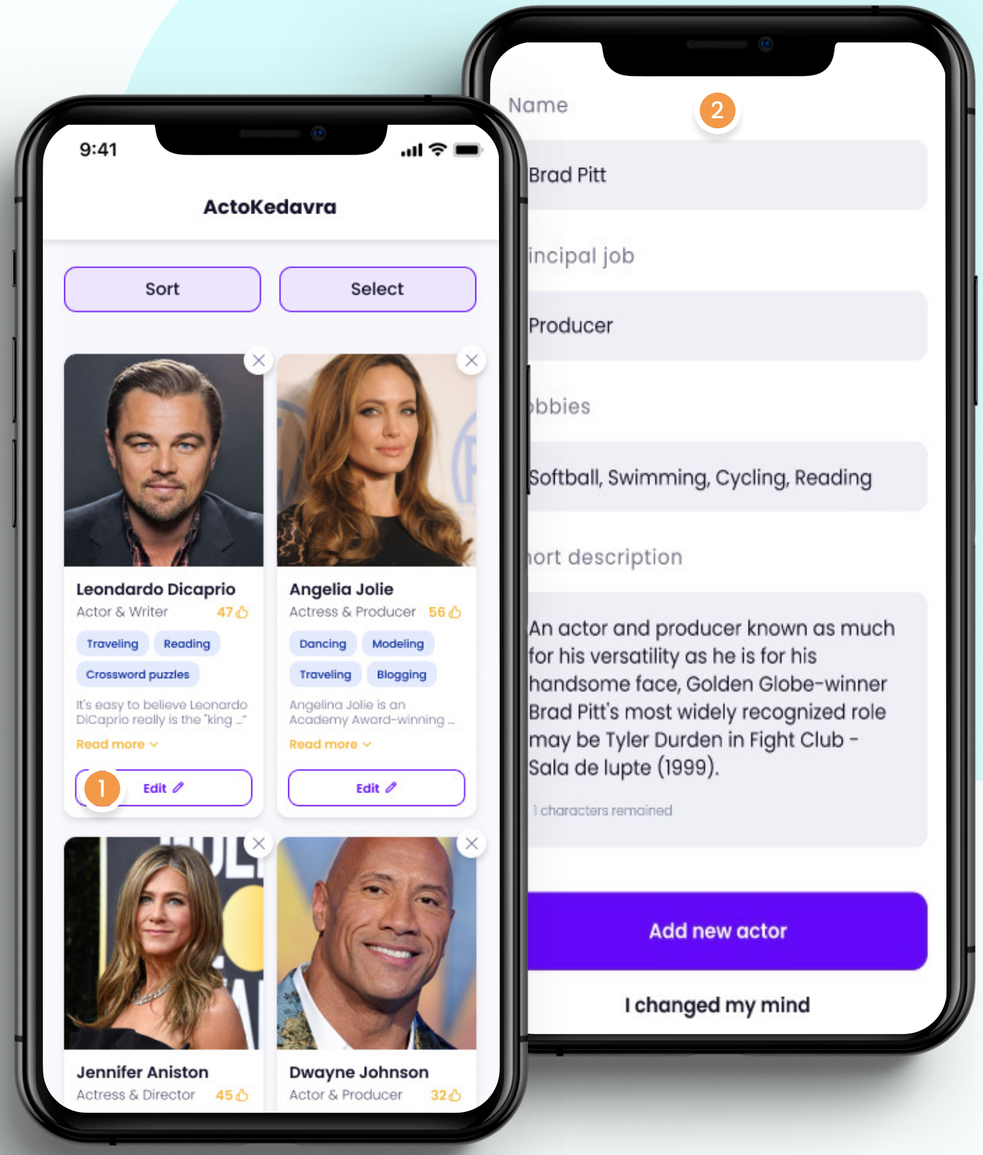
Add an actor

Use the form you've created in the phase of the components, connect it with your list using useState hook and implement the ability to add new actors to your list.



Edit an actor

Use the form you've created in the phase of the components, connect it with your list using useState hook and implement the ability to add new actors to your list.



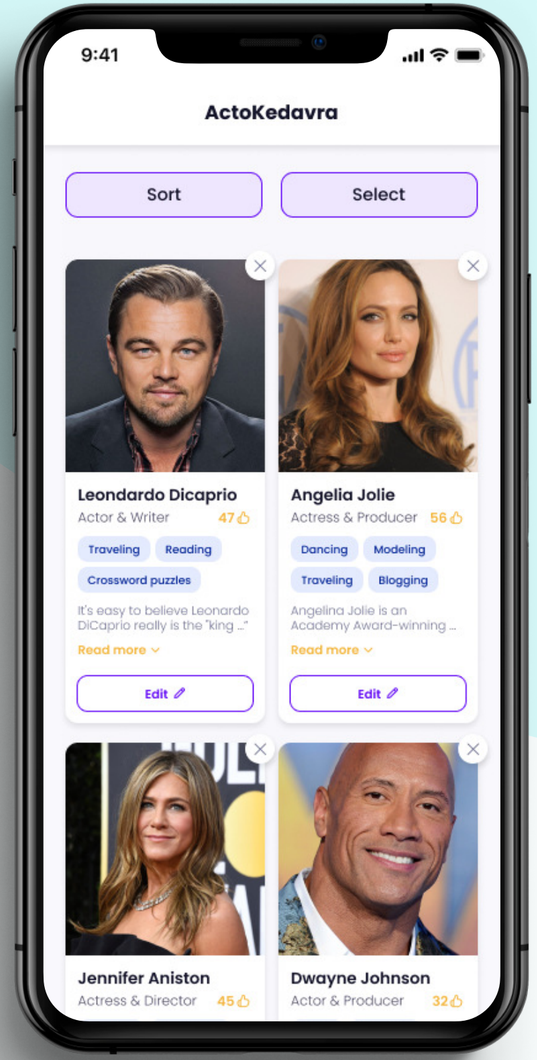
How do I save the image?

When you started connecting the actor form, you probably thought, "I don't see the image input; how am I supposed to add the actor image?". That's a good question. To save an image, you need to store it somewhere in the cloud. An easier way is to convert it to base64 and then save that string in your server-json.

You've stumbled on your first extra task to stretch your skills.

- Create another input that will accept a string representing the image URL.
- Fetch that image with axios and convert it to base64.
- Save the base64 string as part of the actor object when saving the actor into the server-json database.

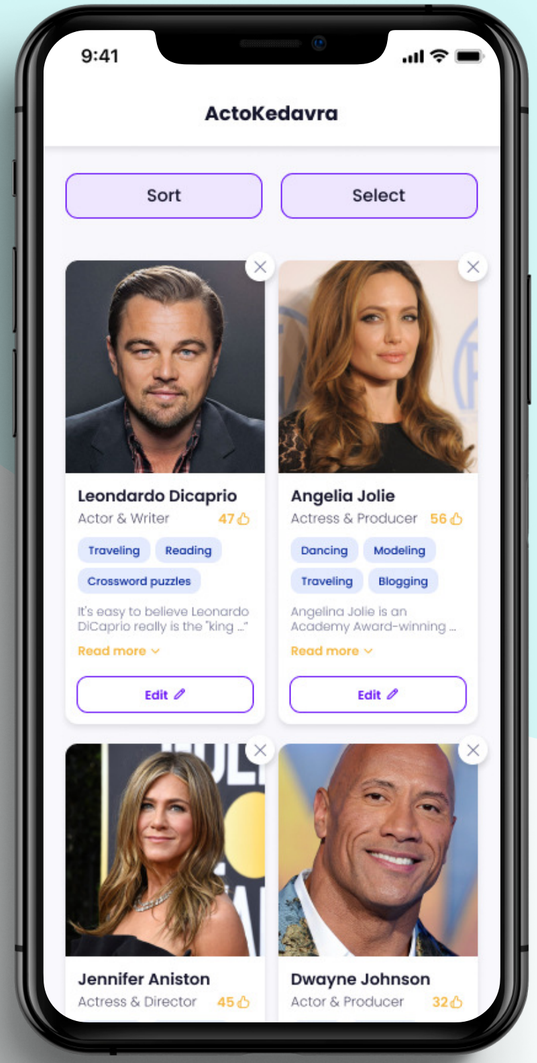
TIP Threat this task as a little extra challenge and do some research on your own.



How do I save the image?

You have some helpful links below.

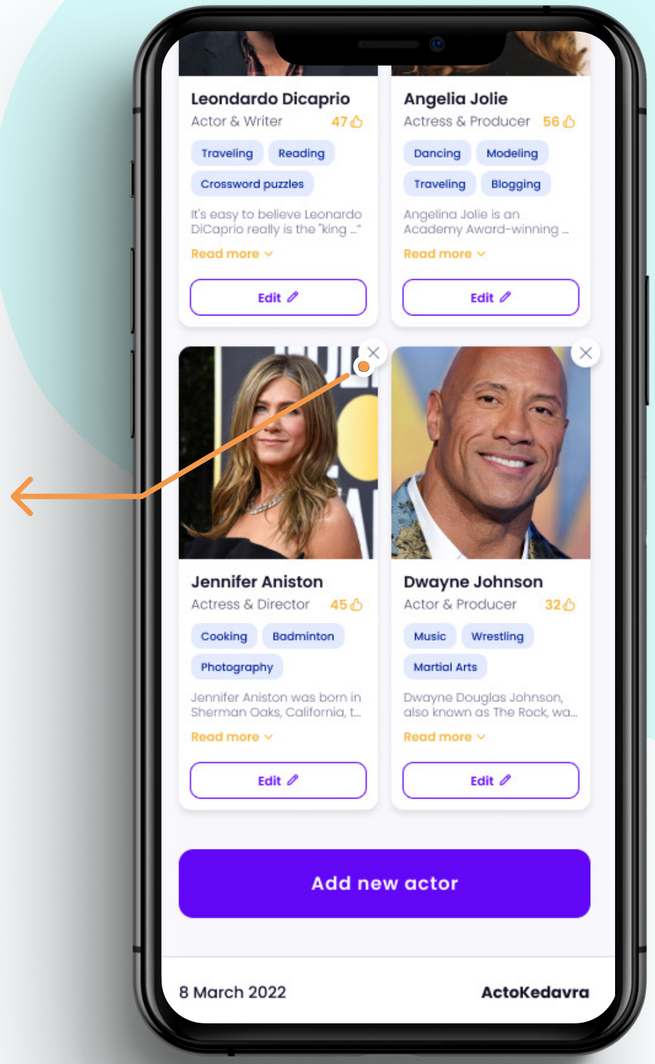
- How To Convert File To Base64 Format React Hook Component
- A funny axios issue



Remove the actor

Display this component only when there are no more actors to show.

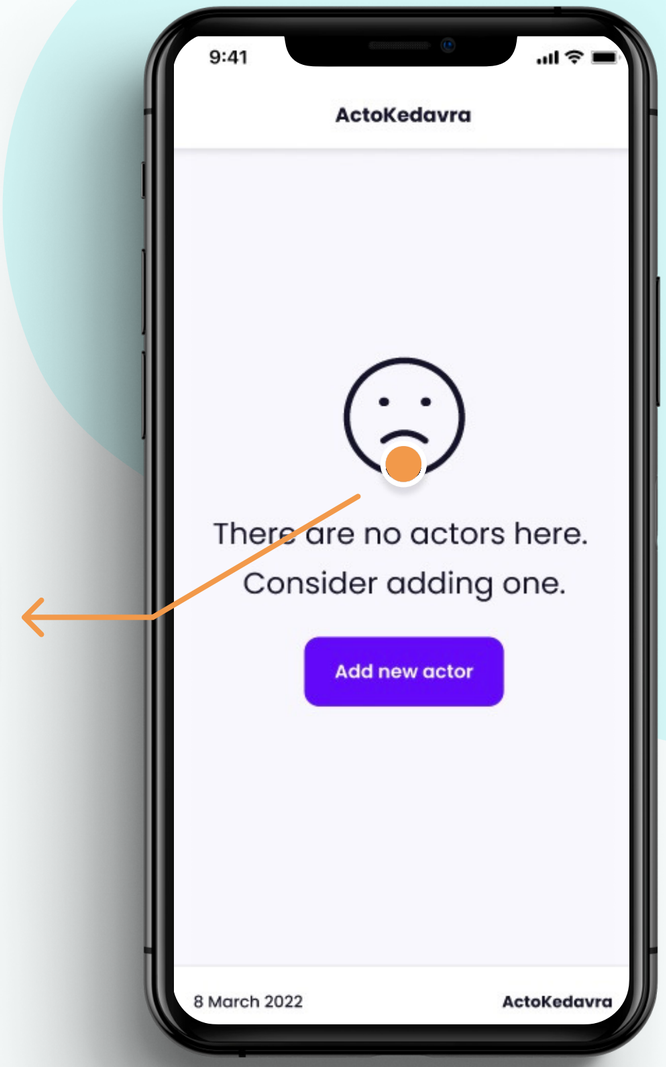
Remove the actor when you click on the x button



Display a call to action when no actors

At this point, you have already implemented the <NoActors /> component. In this task, you should connect the dots and make it so that when there are no more actors to display, you will show the <NoActors /> component as a call to action.

Display this component only when there are no more actors to show.

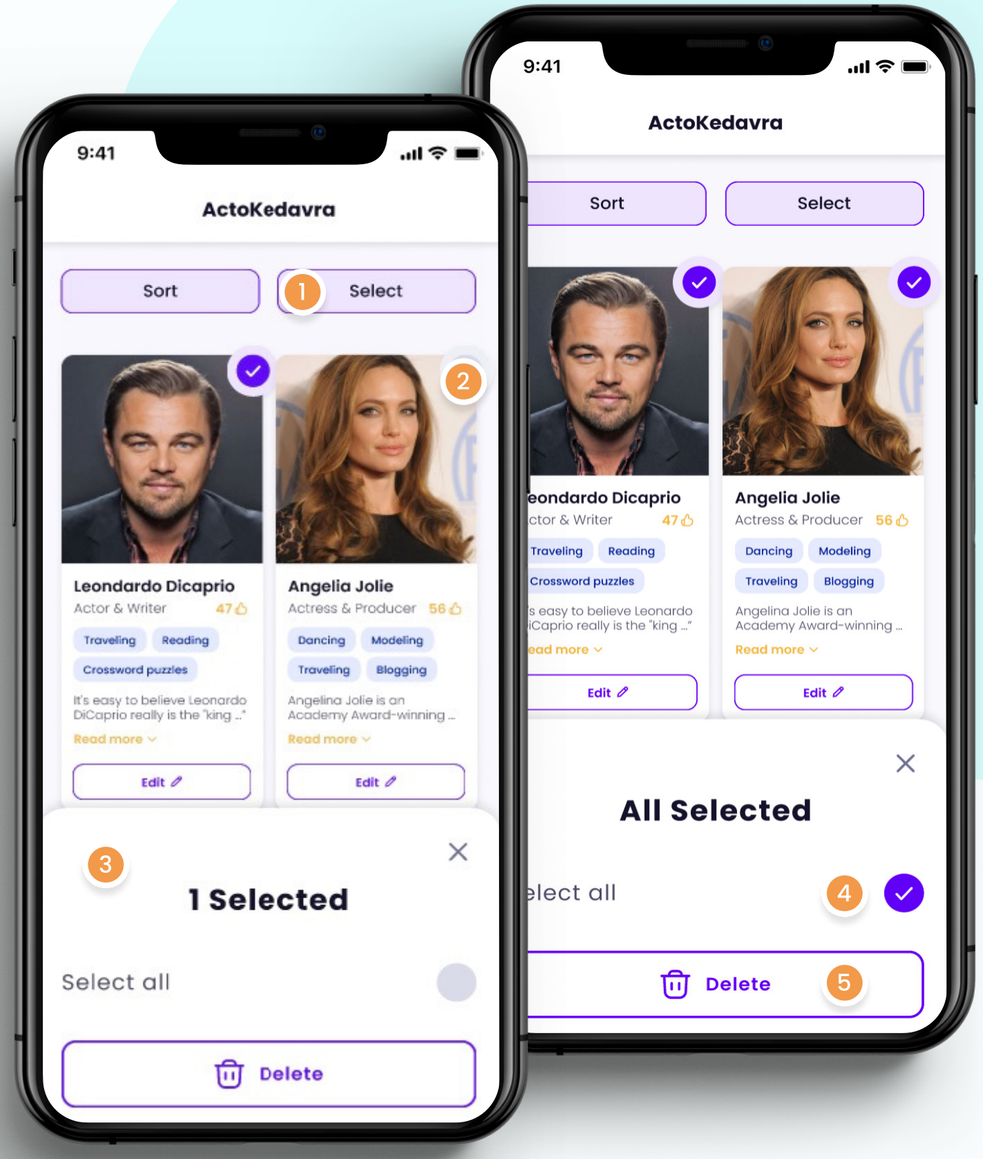


Remove multiple actors

While removing actors one by one provides you with enough fun to make it easier to remove all the actors, you should implement the following functionality.

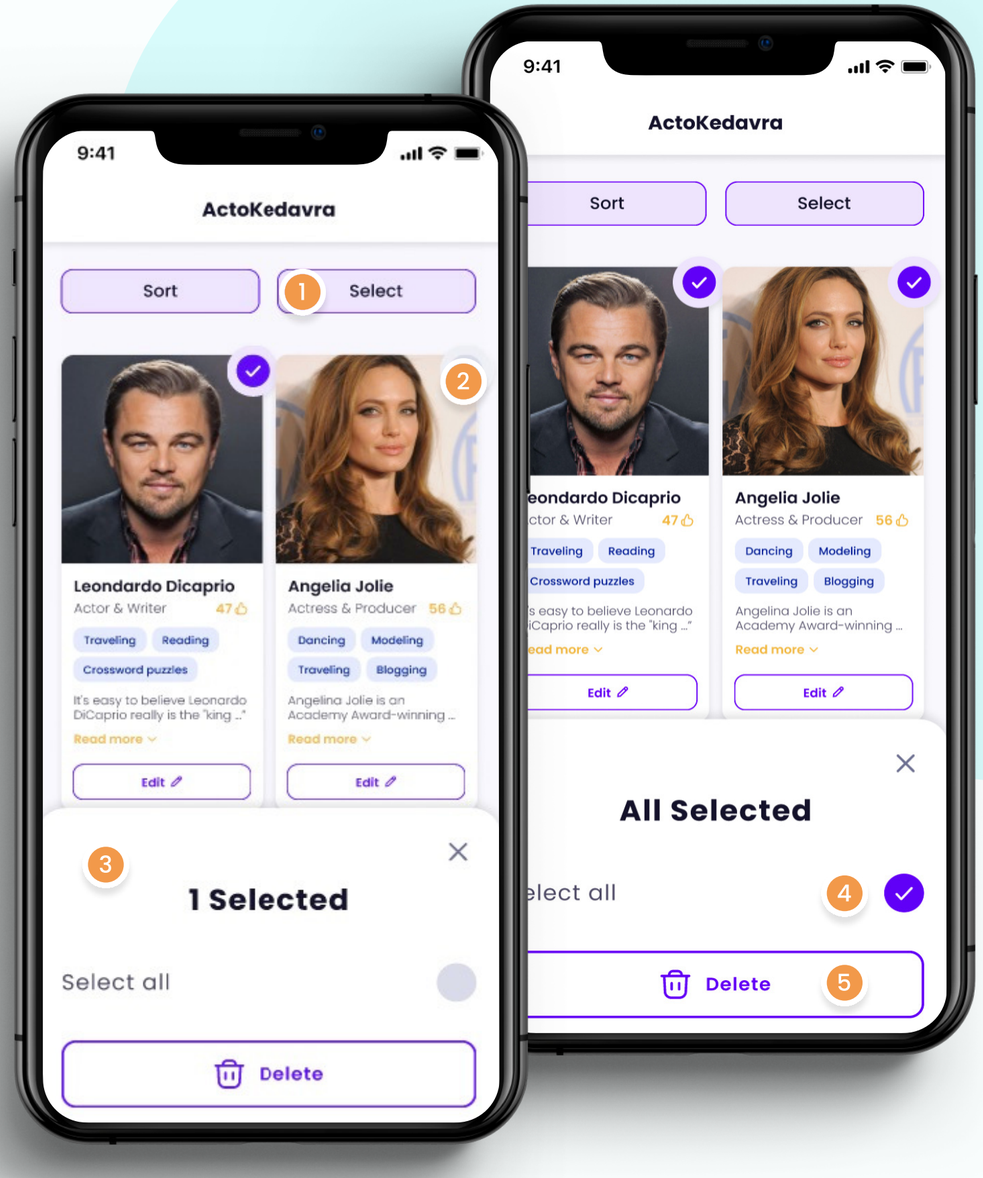
- After you click the "Select" button, display a modal with the option to select all the actors.
- Display a checkbox on each actor card that will allow you to select only a few actors.
- In this select mode, if you have at least one selected actor, enable the delete button from the modal to delete all the selected actors.

If there is something else you think is worth doing for this feature, do it.



Remove multiple actors

1. Click the “Select” button
2. Select at least an actor
3. The modal is visile right after step 1
4. Select all the actors
5. Click the “Delete” button

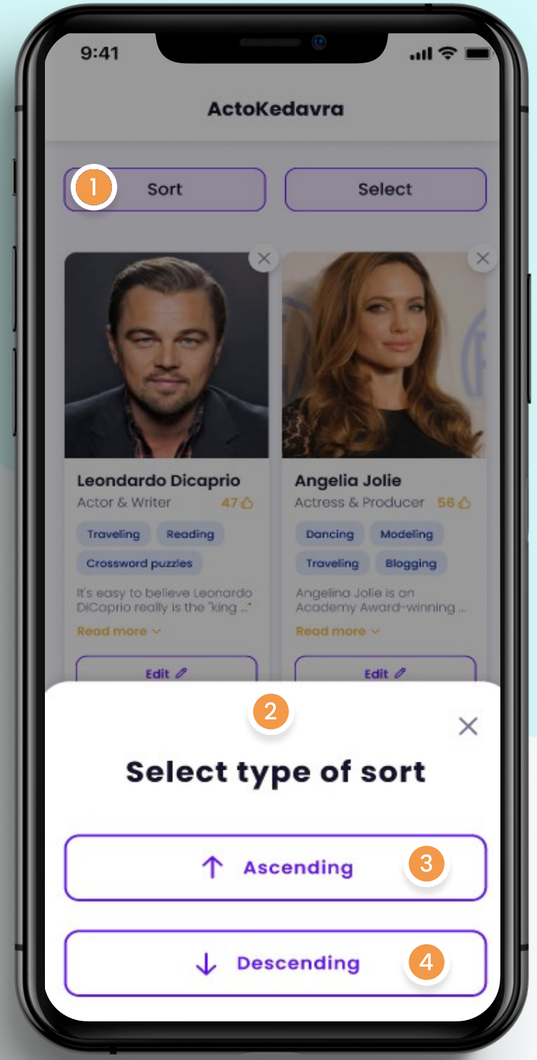


Sort them good, like you should

Another valuable piece of functionality is sorting the actors.

- After you click the "Sort" button, display a modal with the sorting options.
- Do the sorting based on the actor's rating score.

1. Click the “Sort” button
2. Display the modal
3. Sort ascending by score
4. Sort descending by score



TODO, show project structure

07

UI RESPONSIVENESS

TODO

Be Responsive

TODO