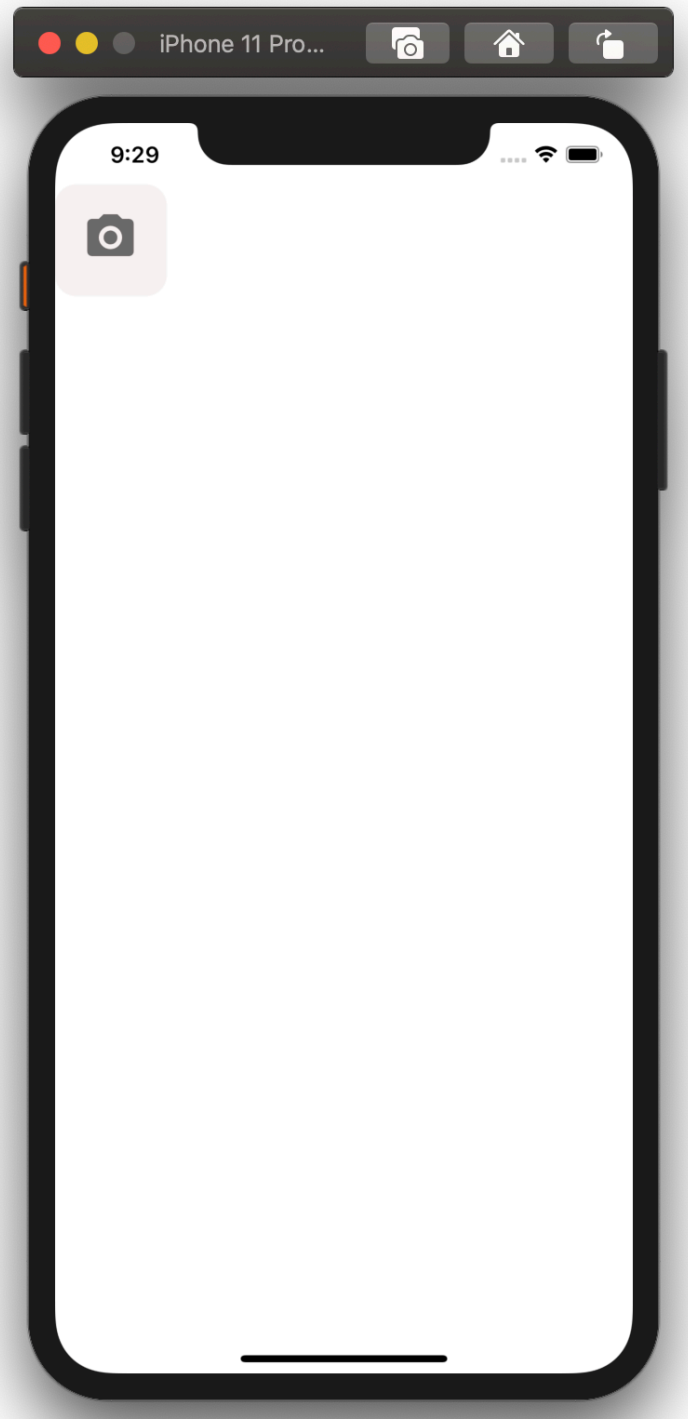
Native Features Exercises

Total exercise time: 60 minutes

ImageInput Component

Exercise time: 20 minutes

Build and test the **ImageInput** component in isolation. This component allows selecting a single image and can be used as follows:




```
<ImageInput
      imageUri={imageUri}
      onChangeImage={(uri) =>
setImageUri(uri)}
/>
```

Props

imageUri: The URI of the image to display. If URI doesn't exist, it should render a camera icon and allow the user to select an image.

onChangeImage: The event that gets raised when the user selects an image.

State

This component doesn't maintain a local state. It's a dumb, presentational component. The state will be maintained by the consumer of this component. So, if we use this component in the **App** component, we need to declare a state variable in the **App** component and pass it as a prop to **ImageInput**:

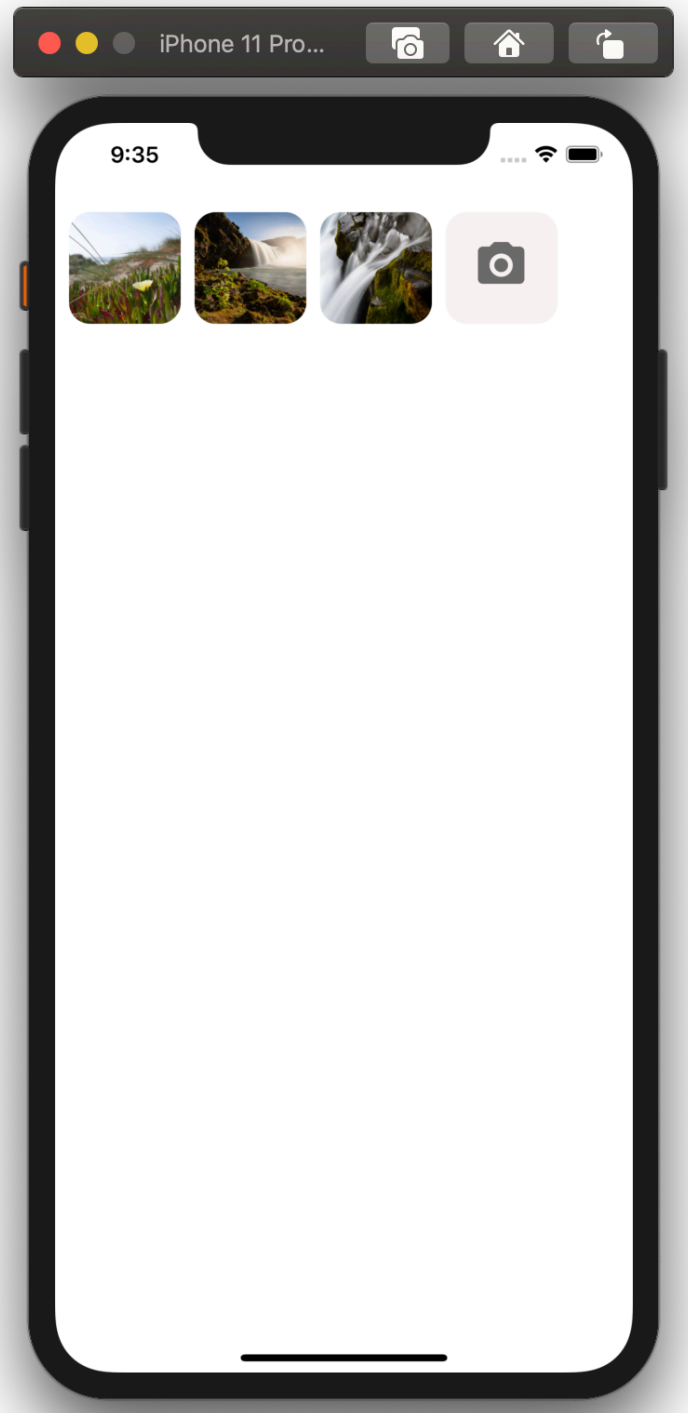
```
const [imageUri, setImageUri] = useState();

<ImageInput
    imageUri={imageUri}
    onChangeImage={uri => setImageUri(uri)} />
```

ImageInputList Component

Exercise time: 10 minutes

This component allows selecting multiple images. It renders the list of images passed to it as a prop and adds the camera icon at the end for selecting a new image.




```
<ImageInputList
      imageUris={imageUris}
      onAddImage={…}
      onRemoveImage={…}
/>
```

Props

imageUris: An array of image URIs to display. Each URI should get mapped to an **ImageInput** component.

onAddImage: The event that gets raised when the user selects a new image to add to the list. The consumer of this component should maintain the state (array of URIs) and update it.

onRemoveImage: The event that gets raised when the user removes an image. The consumer of this component should handle this event and update its state accordingly.

FormImagePicker Component

Exercise time: 10 minutes

To use **ImageInputList** on a Formik form, we need to combine it with an **ErrorMessage** inside a new component (**FormImagePicker**) and give them access to Formik context.

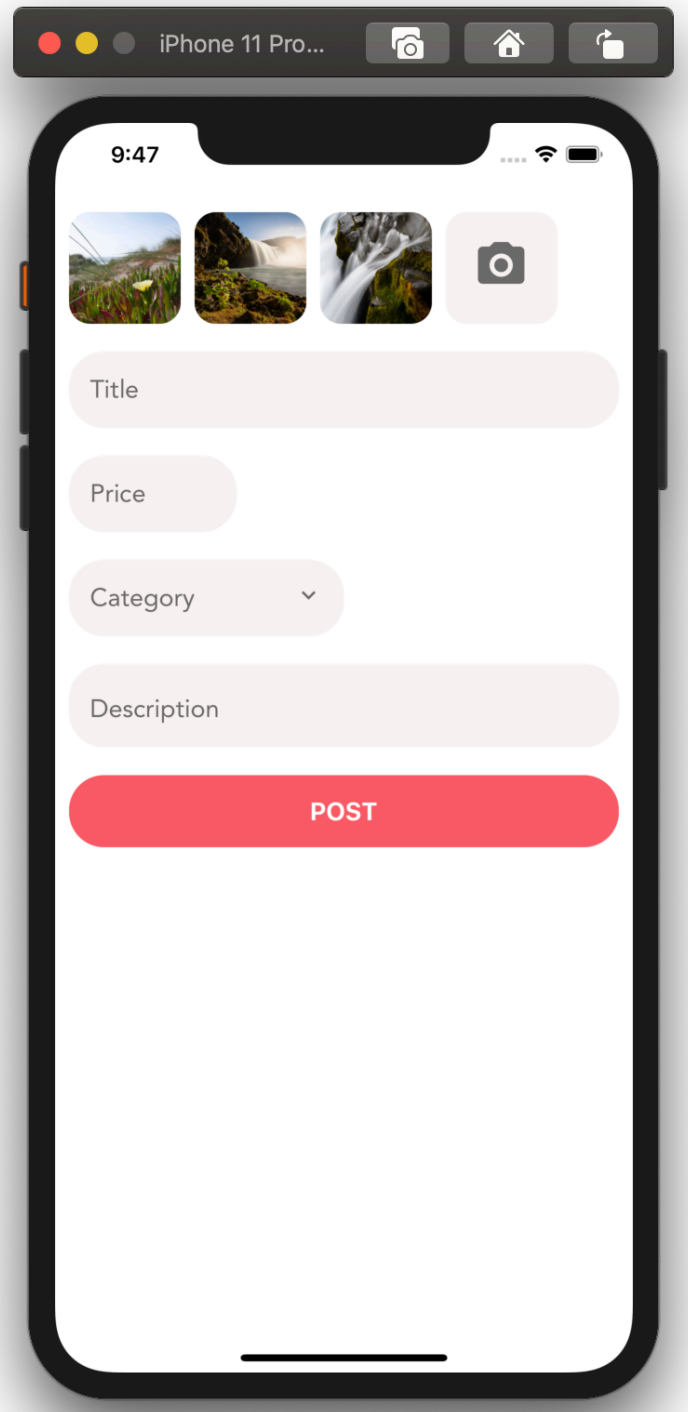
This is the same approach we took in the first part of this course. Earlier, we built two form input components: **FormField** and **FormPicker**. You can look at the code in these components for inspiration.

ListingEditScreen

Exercise time: 10 minutes

Use the **FormImagePicker** component you built earlier and allow the user to add images to a listing. When the form is posted, we should be able to access the URIs of these images just like the other fields of the form.

Make the sure that the user has selected at least one image.



User's Location

Exercise time: 10 minutes

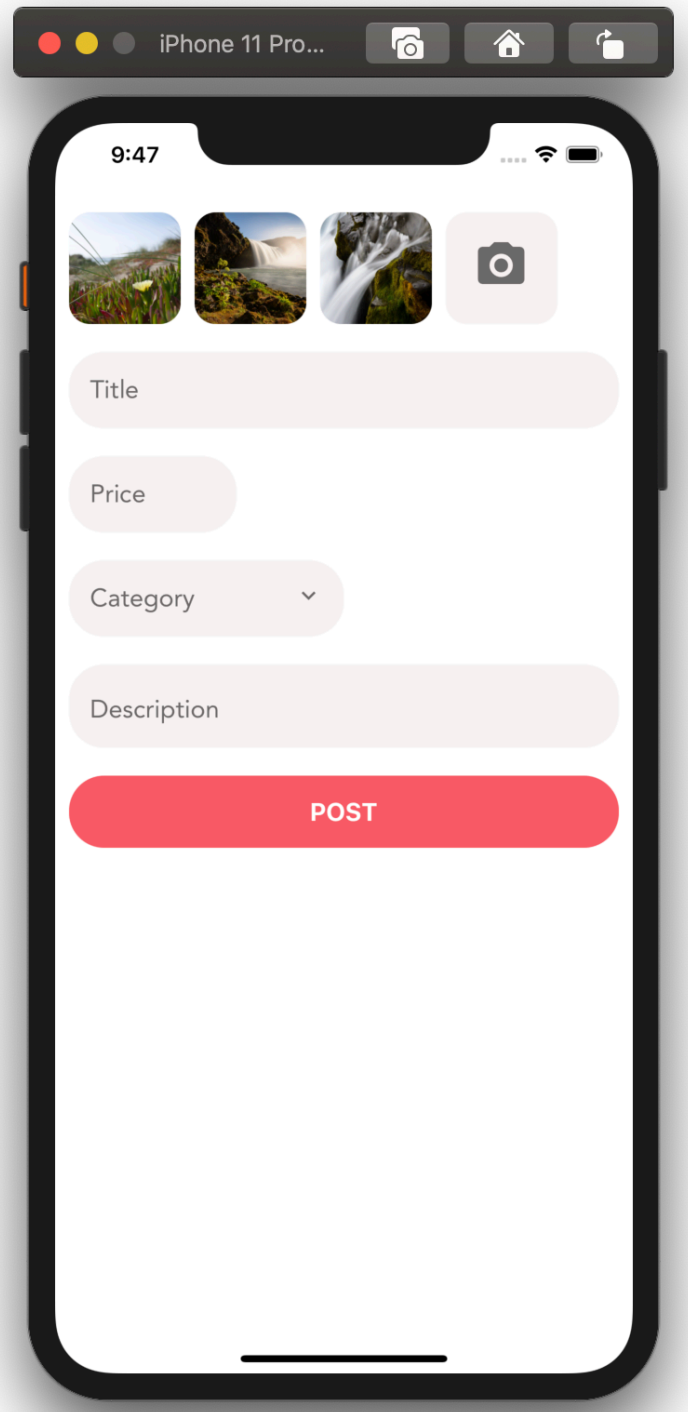
In **ListingEditScreen**, get the user's location using the Location API in Expo.

https://docs.expo.io/versions/latest/sdk/location/

As per the doc, if you're testing the app on a simulator, ensure that location is enabled:

https://docs.expo.io/versions/latest/sdk/location/#enabling-emulator-location

Get the location only once (the first time the screen is rendered) and store it in a state variable. This variable will be available when posting the listing.



Note: The location should be an optional attribute. If the user doesn't give permission to access their location, don't show any error messages. If we have the location of a listing, we'll show it on the map in the future.