## Hands-On - #3

Q1. Make a copy of the 'template' folder and give an appropriate name.

Suggest to name it as 'airbnb'

Open the 'airbnb' folder using VS Code.

[a] Copy the 'airbnb-logo.png' and 'photo-grid.png' files inside the 'images' folder in the 'public' directory

NOTE: The 'images' folder needs to be created

[b] Copy the 'index.html' and 'style.css' files in the 'public' dir.

[c] Build the 'App' component

[d] Build the 'Navbar' component. It should display the 'airbnb' logo

[e] Since the 'Navbar' component has only one element 'nav', let's style it as follows:

```
height: 70px;
display: flex;
background-color: blue;
padding: 20px 35px;
box-shadow: 0px 2.98256px 7.4564px rgba(0, 0, 0, 0.1);
```

NOTE: Properties are taken from 'figma' file.

[e] Comment out the 'background-color' property.

Save all the files and observe the React app.

NOTE: If you are using a 'Safari' web-browser, the 'airbnb' logo could be spread.

To fix that, do the following:

- Give a name to the <img> element in the Navbar component. 'nav--logo'
- In the style.css file set the 'max-width' property to 100px

Q2. The next challenge is to build the 'Hero' component which will have the photo-grid, the 'h1' text and the 'p' (paragraph) text.

This part of the app uses, 'Popping' font. That has been included in the 'index.html' file.

The 'photo-grid' image is centered where as the 'h1' and 'p' elements are right-aligned.

The <h1> element has text 'Online Experiences'

The <p> element has the following text:

Join unique interactive activies lead by one-of-a-kind hosts-all without leaving home.

[a] Create the 'Hero' component with the <section> tag having the image, `h1` and `p` elements

[b] Include the 'Hero' component in the 'App'

[c] Save the files and check your app.

Record your observation: - Particularly the image

[d] Let's style our Hero component elements.
Before we do that let's give the following names:

hero, hero--photo, hero-header and

hero-text to the appropriate elements

- Set the 'section' padding to 20px
- Set the '.hero' and '.hero-photo' properties as follows:

```
.hero {
  display: flex;
  flex-direction: column;
}

.hero--photo {
  max-width: 400px;
  align-self: center;
}
```

- Change the font-family property for body as follows:

```
font-family: 'Popping', sans-serif;
```

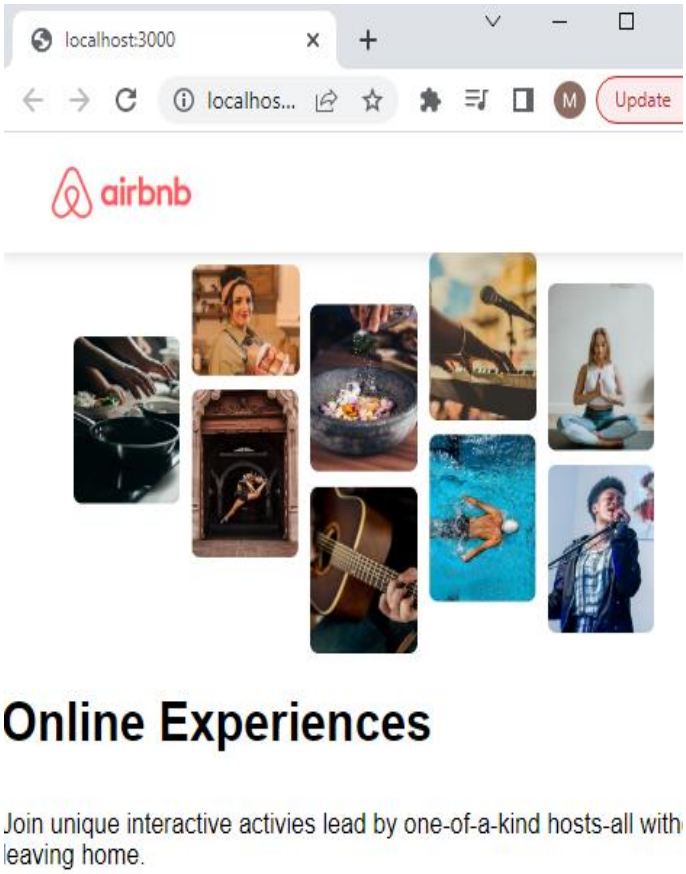
- Set the margins of header and text elements of Hero component as follows:

```
.hero--header {
  margin-bottom: 16px;
}

.hero--text {
  margin-top: 0;
}
```

- Save your CSS file and observe the app.

The output should resemble something like the one show below:




Q3. Challenge: Build the Card component

For now, hard-code in the data (like the rating, title, price, etc.)

Notes:

- Only render 1 instance
- The star icon and photo (katie-zaferes.png) should be in the images folder
- Make sure to include:

  * image
  * star icon (star.png), rating, and review count
  * title
  * cost/person

- The main purpose of this challenge is to show you where our limitations currently are, so don't worry about the fact that you're hard-coding all this data into the component.

[a] Create a 'Card' component which right now display a single card with the image, status and the some paragraphs

```
<div className="card">
    <img src="images/katie-zaferes.png" alt=""
            className="card--image" />
     <div className="card--stats">
         <img src="images/star.png" alt=""
             className="card--star" />
         <span>5.0</span>
         <span>(6) • </span>
         <span>USA</span>
     </div>
     <p>Life lessons with Katie Zaferes</p>
     <p>From $135 / person</p>
 </div>
```

- Comment the 'Hero' component in the app to see the progress

- Save your file and observe the app.

[b] Let's provide the styling for the card as follows:

```
.card {
    width: 175px;
font-size: 12px;
}

.card--image {
width: 100%;
border-radius: 9px;
}

.card--stats {
display: flex;
align-items: center;
}

.card--star {
height: 14px;
}
```

[c] The color for the text in the stats part is 'gray'

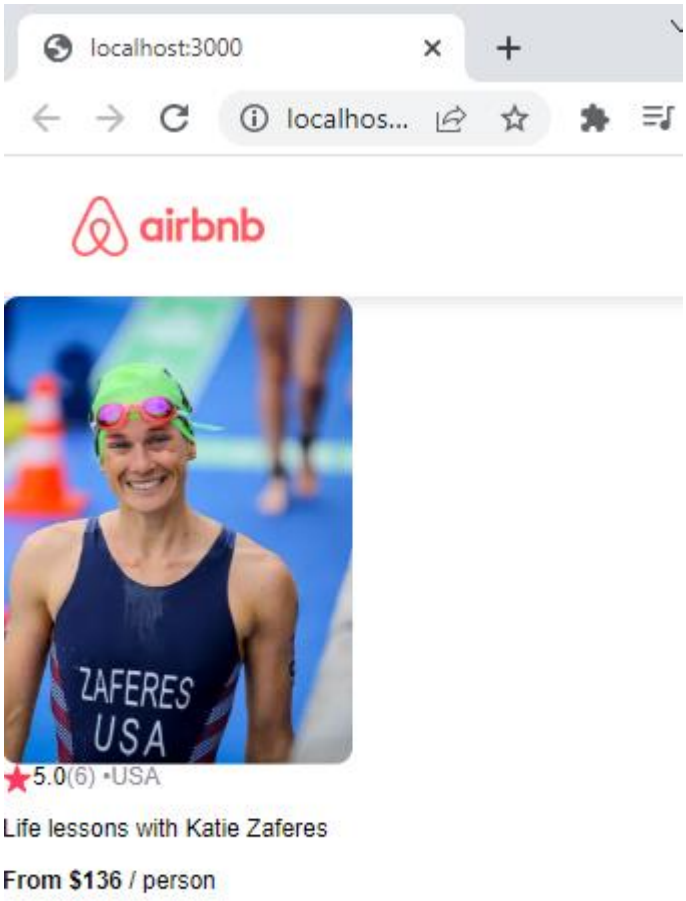
```
.gray {
color: #918E9B;
}
```

- Also observe that the first part of the cost/person is bold.

  Style accordingly.

- Save your files and observe the app.

The output should look as shown below




Q4. Make a copy of the 'template' folder and this time call it as 'catsapp'

- Open 'catsapp' in VS Code

[a] Unzip the 'LABp2q4' file and copy as follows:

- The 'images' folder inside the 'public' directory.
- The 'style.css' and 'index.html' files inside the 'public' dir.
- The 'index.js' and 'App.js' files inside the 'src' dir.

[b] Open the 'Terminal' window and start your app.

Record your observation.

[c] Now your challenge is as follows:

- Create a Contact.js component in another file

- Move one of the contact card div's below into that file

- import and render 4 instances of that contact card

- Think ahead: what's the problem with doing it this way?

[d] Let's pass value(s) to over Contact component
- Done similar to setting the attributes in HTML
- Since we have 4 instances of the contact component
pass the values: img, name, phone and email
- Save and check the app.

NOTE: Although we have passed the values, but the Contact component is not using them.

- console.log to see if we are receiving the values.

You can give whatever name you want the values you receive

[e] The right value to name the value(s) for a component is - 'props' (short for properties)

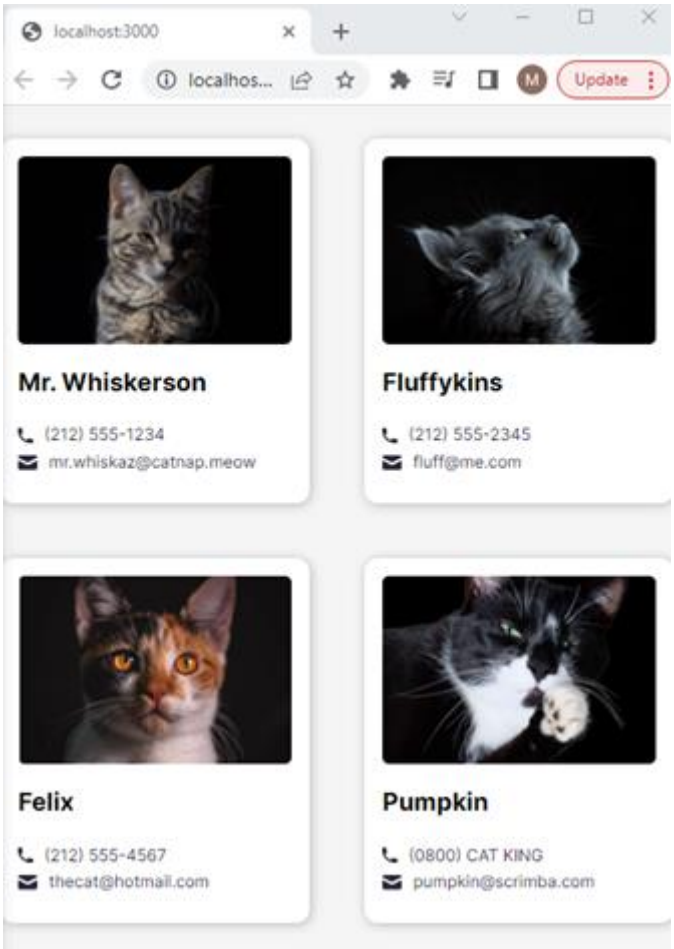
- Fix the code in contact component to use the `props` obj data in appropriate places

- Once again save the files and check the app.

[f] De-structure the props object to appropriate names in the contact component.

- Use the de-structured prop object data and fix the Contact component.

- Save the file(s) and check the app.




Q5. Reopen the earlier project folder 'airbnb' in VS Code.

[a] Your Next Challenge!
Pass props to the Card component and display that data

- img ("katie-zaferes.png")
- rating ("5.0")
- reviewCount (6)
- country (Whatever you want)
- title ("Life Lessons with Katie Zaferes")
- price (136)

[b] Make necessary changes to the App component to pass the data and the Card component to receive the 'props' and use them.

[c] Save your files and check your app.

The output will be similar to the one which is there with Q3.