Initializing the Application

This lesson helps you set up the React environment required to build our app.

WE'LL COVER THE FOLLOWING

Loading the Framework

Let's get started with building our application using **React** and **Firebase**!

The application we are going to build can be set up with **Facebook's official React boilerplate** project - `create-react-app`. You can install it globally using the command line or terminal by following the code snippet given below, after which it becomes available whenever you need it.

```
npm install -g create-react-app
```

After you are done with the installation, you can create your project using `create-react-app` and name it whatever you want. In this course, we are naming the project `react-firebase-authentication`. You can navigate into the project folder using the `cd` command.

```
create-react-app react-firebase-authentication
cd react-firebase-authentication
```

You can use the following command on your command line to start your application, after which you will be able to access it in the browser.

```
npm start
```

Now, we'll set up the project according to our requirements. First, we will get rid of files that were automatically downloaded from the *boilerplate React project* as we won't be using them. On the command line, navigate to your **src**/ folder using the `cd` command and then use the `rm` command to remove the following files from your folder.

```
cd src
rm App.js App.test.js App.css logo.svg
```

Second, create a folder and name it **components** in your application's **src** folder using the `mkdir` command on the command line. We will implement all the components in this folder including the *App component* that we removed in the previous step.

```
mkdir components
```

Now that you have created the **components** folder, navigate into it using the `cd` command and create a dedicated folder for each component that we will be implementing for this application.

There are a total of **13 components** that we will be implementing for this application and they are as follows:

- Account
- Admin
- App
- Home
- Landing
- SignIn
- SignOut
- SignUp
- Navigation
- PasswordChange
- PasswordForget

- Session
- Firebase

Hence, a folder will be created for each of them. For the sake of readability, we have divided the commands into multiple lines:

```
cd components
mkdir Account Admin App Home Landing SignIn SignOut SignUp
mkdir Navigation PasswordChange PasswordForget
mkdir Session Firebase
```

In each folder, we will create an `index.js` file for the component. To do this, follow the instructions below:

1. Navigate into the **components** folder using the `cd` command
2. Create the `index.js` file using the `touch` command
3. Navigate out to the **components** folder using the `cd..` command.

Repeat these steps for each component as shown in the code snippet below. Also, note that you can choose to name your *folders or files* differently, but these are the names we will be using in this course.

```
cd App
touch index.js
cd ..
```

Next, we will implement a basic **React component** for each *component file* that we created. For example, for the **App component**:

- Navigate into the `src/components/App` directory
- Open the `index.js` file

The file will probably look like this:

```
import React from 'react';

const App = () => (
  <div>
    <h1>App</h1>
  </div>
);

export default App;
```

App/index.js

There is an `index.js` file in your original **src** folder. Open that file and fix the *relative path* to the **App component** since you moved the **App component** to the `src/components` folder. To fix the path, you need to add the `/components` subpath to it as you can see in the code snippet below.

```
import React from 'react';
import ReactDOM from 'react-dom';

import './index.css';
import * as serviceWorker from './serviceWorker';

import App from './components/App';

ReactDOM.render(<App />, document.getElementById('root'));

serviceWorker.unregister();
```

Then, create one more folder, **constants**, in your **src** folder as shown in the code below.

```
mkdir constants
```

The folder will be located next to the **components** folder in the **src** directory. Now, navigate into the **constants** folder and create 2 files: First, for the application's routing `routes.js`; Second, for the roles management `roles.js`.

```
cd constants
touch routes.js roles.js
cd ..
```

Congratulations! Your application with all its folders and files is now set up. Go to the next lesson to verify and run the app on our live terminal!