Students Information

Name	Class ID	Email
Corey Doss	8	cjdw94@mail.umkc.edu
Khalid Dhabbah	6	kmdk2t@mail.umkc.edu

Introduction

For this lab exercise 3, we will use API files again, and then we are going to search for a specific term, which is the **whole chicken**. To do so, we should sign up in Nutritionix's and IBM's sites, so that we could have our keys and use them in our project. The new goal is to create a login form, where no one should open the page of nutrition and IBM until the sign-in function is done. Besides, we are going to use bootstrap and jquery in our pages to get some great features like letting the site adjust in a variety of screens.

Objectives

1- Installation

- Install bootstrap.
- Install jquery.
- Install popper.
- Install font-awesome.

2- Creation

- Create a header component.
- Create a home page.
- Create our page for nutrition and IBM.
- Create the login & register page.

Design/Implementation

First, in order to install all the requirements, we are going to use NPM. After we create a new project using Angualrjs CLI in WebStorm, we use the Terminal section for writing NPM commands. To install bootstrap, jquery and popper, we are going to use the following commands.

```
$ npm install bootstrap
$ npm install jquery poppoer.js --save
```

Then we are going to open angular.json to add them there. After finding styles and scripts, we replace them by the following code.

```
   "styles": [
              "src/styles.css",
              "./node_modules/bootstrap/dist/css/bootstrap
.min.css",
              "./node_modules/font-awesome/css/font-awesom
e.min.css"
            ],
```

```
            "scripts": [
              "./node_modules/jquery/dist/jquery.slim.min.
js",
              "./node_modules/popper.js/dist/umd/popper.mi
n.js",
              "./node_modules/bootstrap/dist/js/bootstrap.
min.js"
            ],
```

Now, we are done from the installation, and we can use bootstrap in our project and make some stylish.

Second, we want to create a new component for the header. The reason for doing it is to divide our project into understandable sections, and thus, it is going to be easy to make some changes later.
To do so, we are going to use the next command in the Terminal.

```
$ ng generate component header
or
$ ng g c header
```

This command will help us create a folder called header, which contains the important files we need. The next step is to add our new component to other components and to add it; we could reference it by **app-header**.

```
<app-header></app-header>
```

Now, it is time to create the other components:

```
$ ng generate component home
$ ng generate component login
$ ng generate component register
or
$ ng g c home
$ ng g c login
$ ng g c register
```

Third To link our pages together, we have to create a routing module by using the following coming and do the configuration.

```
$ ng generate module app-routing --module app --flat
```

We import this routing module into our **app-module**.

Input/Output

Conclusion