SIT 725 Prac 3
Bootstrapping Express App

Contents

- Bootstrapping Express App
- Adding some basic html code
- Basics of Materialize
- Advance Components Materialize
- Conclusions
- Questions

Bootstrapping Express App

In order to start with bootstrapping the express app you first need to create a github repo which we discussed in last practical, please follow the same instructions which we did during the practical.

You would also need to have nodejs installed before this step.

Once you have created the github repo, clone the repo in your system

```
$ git clone <link to repo>
```

After you have done that move into the folder

```
$ cd sit725-2021-t2-prac3
```

Bootstrapping Express App Cont ...

Now that you are inside the folder we can start bootstrapping the express app

First step is to initiate a node environment inside your repo for doing that run the command

```
$ npm init
```

Once you have run the command it would ask you a few question which you can answer and it would create a file called package.json inside your application.

Next we add express to our application, for that we run the command

```
$ npm install express --save
```

Bootstrapping Express App Cont ...

Doing this create a folder called node_modules in your application and will also update the package.json.

Now we are ready to create our first express app.

Next we create a file called server.js in our application, this file will be responsible for creating our server and handling all the process for our application.

Your server.js should look like this:

```
var express = require("express")
var app = express()
var port = process.env.port || 3000;

app.listen(port,()=>{
    console.log("App listening to: "+port)
})
```

Bootstrapping Express App Cont ...

Once we have done this we need to update our package.json In the scripts section of our package.json we add a new script to start our application. Your package.json should look like this

```
{
  "name": "sit725-2021-t2-prac3",
  "version": "1.0.0",
  "description": "sit 725 week 3 prac",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ChoudharyNavit22/sit725-2021-t2-prac3.git"
  },
  "keywords": [
    "Nodejs",
    "Express",
    "Materialize",
    "HTML",
    "JS",
    "CSS"
  ],
  "author": "Navit Choudhary",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/ChoudharyNavit22/sit725-2021-t2-prac3/issues"
  },
  "homepage": "https://github.com/ChoudharyNavit22/sit725-2021-t2-prac3#readme",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Bootstrapping Express App Cont ...

Please make sure you only update the script section to look the same as example rest should not be changed as they are specific to your own repo the example is from my personal repo so it has the git url and author names from my repo

Once you have done all this you are ready to run your first express app.

In order to do that in your terminal run the command

```
$ npm run start
```

And you should have something printed in your terminal saying that the application is running on port 3000, Now you have your first running Express App. **Please don't forget to commit your code at this step.**

Adding some basic HTML code

Now that we have created our basic express app lets add some basic html code to our application.

So we create a folder inside our application named public and then two more folders inside our public folder named js and css.

Js folder would be the folder that would have our custom javascript code and css folder would be responsible to handle our custom css very similar to how we did in week 1

Now we need to let our express app know that we are going to serve some static HTML, JS and CSS

So in order to do that we modify our server.js file.

Adding some basic HTML code Cont ...

Now our server.js file should look something like this

```
var express = require("express")
var app = express()

app.use(express.static(__dirname+'/public'))
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

var port = process.env.port || 3000;

app.listen(port,()=>{
    console.log("App listening to: "+port)
})
```

Adding some basic HTML code Cont ...

Now our express application knows that we would be serving some status web application from the public folder.

Now let's add some static things to our public folder. Inside our public folder we create a file called index.html and inside our js folder which we created before we create another file called scripts.js and same in our css folder we create a file called styles.css and as we said before these file will be responsible for handling our custom javascript code and css styles. We also created a folder called images inside our public folder from where we will serve our images being used on our web page.

We will also add the cdn links for materialize css and materialize js as we did in week 1 but at this point we are not going to use any of it.

Adding some basic HTML code Cont ...

Now let's add some web code to our index.html

Adding some basic HTML code Cont ...

Now when we run our application this is what our web page looks like.

Pretty Ugly right, Can't agree anymore. Now let's try to make it look much better by using our friend Materialize

Basics of Materialize

The first thing we do is we add a navbar to our application. So we got to materialize website and search for a navbar at it give a very nice code that we can use directly with a little modifications. Now our index.html looks something like this.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>SIT 725 Prac 3</title>
  <meta name="description" content="basic template for SIT 725 Prac 3">
  <meta name="author" content="SitePoint">
  <meta property="og:title" content="SIT 725 Prac 3">
  <meta property="og:type" content="website">
  <meta property="og:description" content="basic template for SIT 725 Prac 3">
  <!-- Compiled and minified CSS -->
  <link rel="stylesheet"
ref="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
  <!-- Custom Style Sheet-->
  <link rel="stylesheet" href="css/styles.css">

</head>
<body>
    <nav>
        <div class="nav-wrapper navbar-size">
          <a href="#" class="brand-logo"><img class="user-image-size" src="images/user-image.png" /></a>
          <ul id="nav-mobile" class="right hide-on-med-and-down">
            <li><a target="_blank" href="https://www.linkedin.com/in/navit-choudhary-
5133b9104/">LinkedIn</a></li>
          </ul>
        </div>
    </nav>
  <!-- Compiled and minified JavaScript -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
  <script src="js/scripts.js"></script>
</body>
</html>
```

Basics of Materialize Cont ...

We also added some custom css classes for our user image and navbar width. So our styles.css looks like this.

```
.navbar-size {
    width: 95%;
    margin: 0 auto;
}
.user-image-size {
    height: 64px;
}
```

Basics of Materialize Cont ...

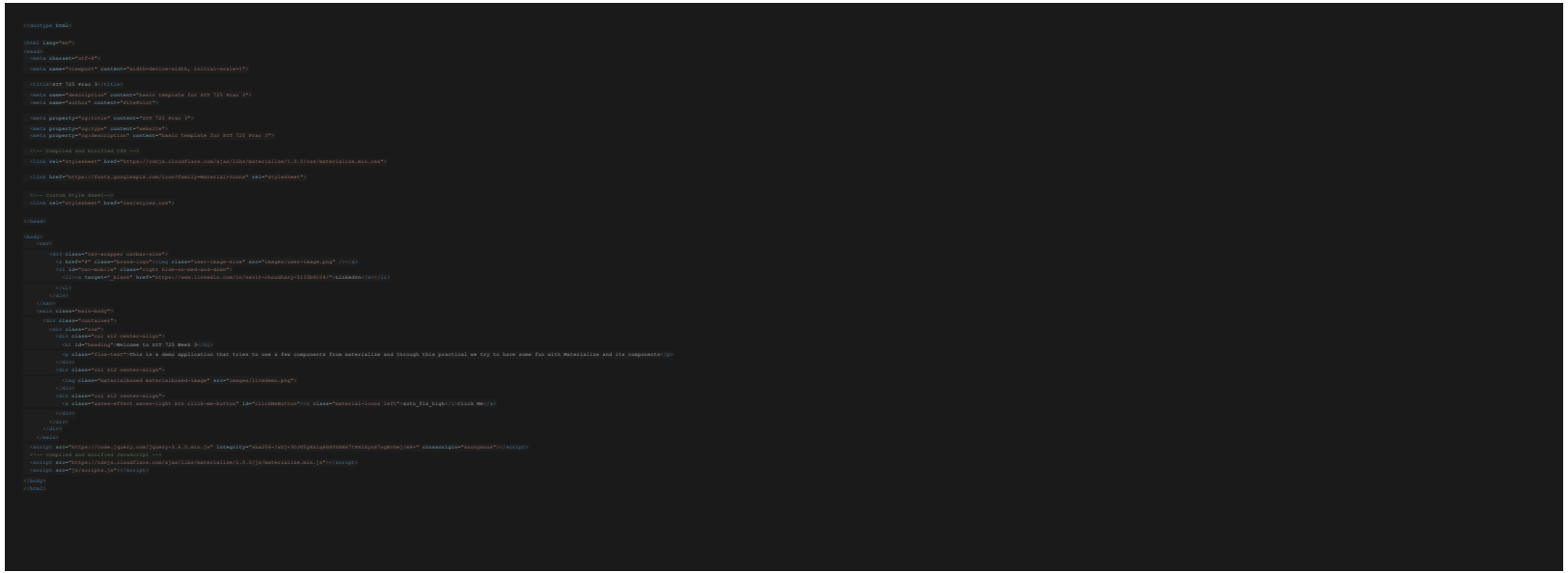
Now let's see how our application looks like. So let's run our application again.

Looks much better already.

Now lets see if we can add more components to it and make our web page look much better.

Basics of Materialize Cont ...

So we add some components from the Materialize website and now our index looks like this.



Basics of Materialize Cont ...

We also updated our styles.css and scripts. So our style.css looks like this

```
.navbar-size {
    width: 95%;
    margin: 0 auto;
}
.user-image-size {
    height: 56px;
    margin: 4px 0;
}
.main-body {
    background-color: #F28487;
    color: white;
    height: calc(100vh - 64px);
}
.materialboxed-image {
    margin: 0 auto;
    height: 400px;
}
.click-me-button {
    background-color: #EE6E73;
    margin-top: 50px;
}
```

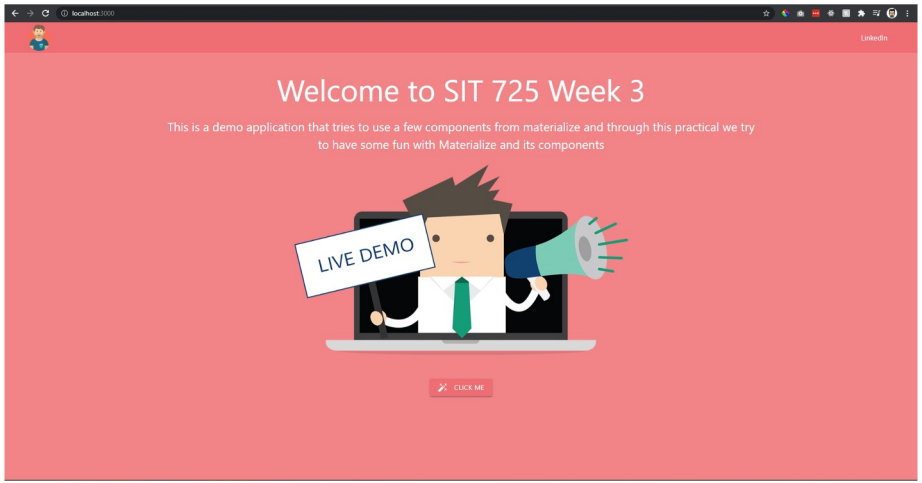
Basics of Materialize Cont ...

Also our scripts.js looks like this

```
const clickMe = () => {
    alert("Thanks for clicking me. Hope you have a nice day!")
}

$(document).ready(function(){
    $('.materialboxed').materialbox();
    $('#clickMeButton').click(()=>{
        clickMe();
    })
  });
```

Basics of Materialize Cont ...

Lets run and see how our application looks like now.



Looks much better from what we had originally right. **Do try to click the button and image**. Also don't forget to commit your code

Advance Components Materialize

Now that we created a very basic static web page lets try to add some new components to our web application.

First we will add a very popular components called **Cards.**

I have attached a link where you can find how to add cards to our application. Lets try adding them now.

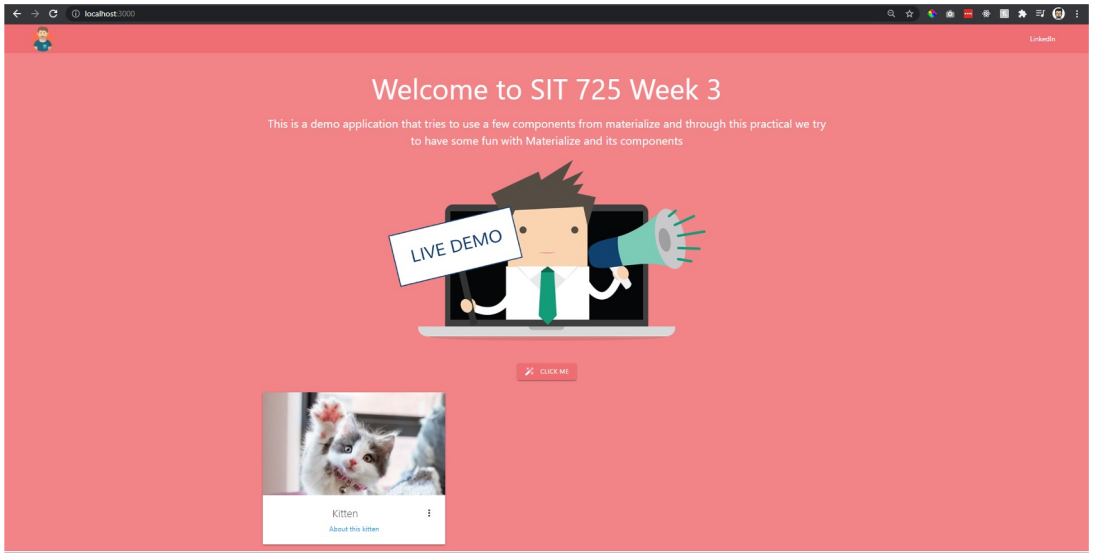
Advance Components Materialize Const ...

Now the body section of our index.html looks like this.

```
<body>
    <nav>
        <div class="nav-wrapper navbar-size">
          <a href="#" class="brand-logo"><img class="user-image-size" src="images/user-image.png" /></a>
          <ul id="nav-mobile" class="right hide-on-med-and-down">
            <li><a target="_blank" href="https://www.linkedin.com/in/navit-choudhary-5133b9104/">LinkedIn</a></li>
          </ul>
        </div>
    </nav>
    <main class="main-body">
      <div class="container">
        <div class="row">
          <div class="col s12 center-align">
            <h1 id="heading">Welcome to SIT 725 Week 3</h1>
            <p class="flow-text">This is a demo application that tries to use a few components from materialize and through this practical we try to
have some fun with Materialize and its components</p>
          </div>
          <div class="col s12 center-align">
            <img class="materialboxed materialboxed-image" src="images/livedemo.png">
          </div>
          <div class="col s12 center-align">
            <a class="waves-effect waves-light btn click-me-button" id="clickMeButton"><i class="material-icons left">auto_fix_high</i>Click Me</a>
          </div>
        </div>
        <div class="row">
          <div class="col s4 center-align">
            <div class="card">
              <div class="card-image waves-effect waves-block waves-light">
                <img class="activator" src="images/kitten.jpg">
              </div>
              <div class="card-content">
                <span class="card-title activator grey-text text-darken-4">Kitten<i class="material-icons right">more_vert</i></span>
                <p><a href="#">About this kitten</a></p>
              </div>
              <div class="card-reveal">
                <span class="card-title grey-text text-darken-4">Kitten<i class="material-icons right">close</i></span>
                <p class="card-text">Hello There! I just wanted to say HI to you guys. See ya!</p>
              </div>
            </div>
          </div>
        </div>
      </div>
    </main>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
  <!-- Compiled and minified JavaScript -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
  <script src="js/scripts.js"></script>
</body>
```

Advance Components Materialize Cont ...

Now lets run it and see how our application looks like.



Advance Components Materialize Cont ...

Looks pretty nice, but would we be giving out static data in a real world web application. Not really, so how about we try to add dynamic data and then create cards on the run.

So we update our index.html body

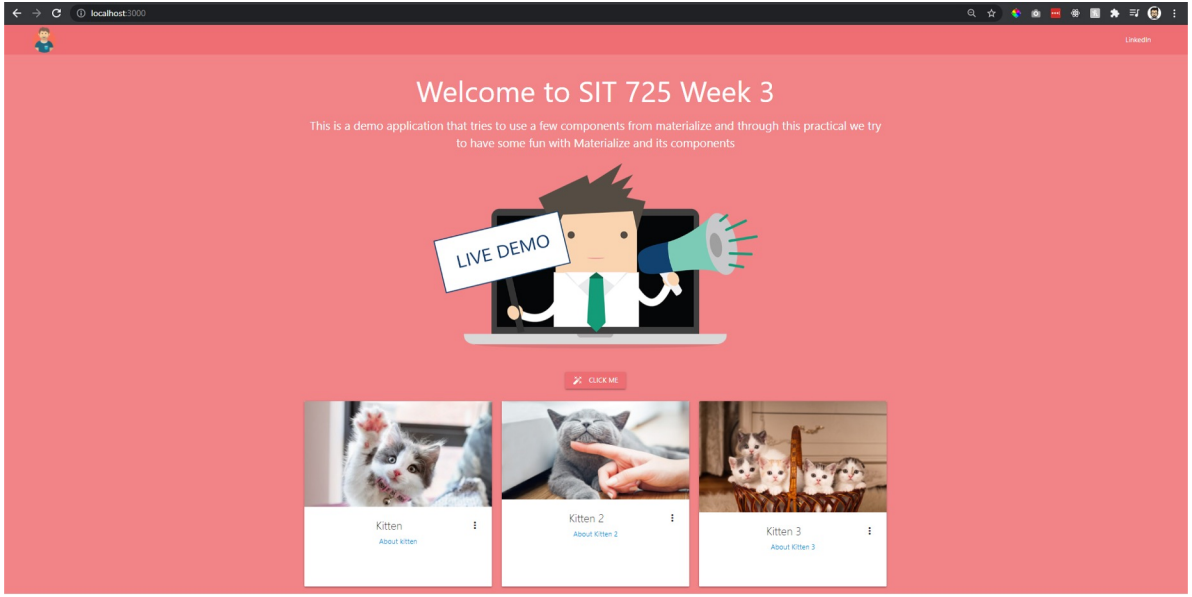
```
<!doctype html>



<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">



  <title>SIT 725 Prac 3</title>
  <meta name="description" content="basic template for SIT 725 Prac 3">
  <meta name="author" content="SitePoint">



  <meta property="og:title" content="SIT 725 Prac 3">
  <meta property="og:type" content="website">
  <meta property="og:description" content="basic template for SIT 725 Prac 3">
```

Advance Components Materialize Cont ...

```
const cardList = [
    {
        title: "Kitten 2",
        image: "images/kitten-2.jpg",
        link: "About Kitten 2",
        desciption: "Demo desciption about kitten 2"
    },
    {
        title: "Kitten 3",
        image: "images/kitten-3.jpg",
        link: "About Kitten 3",
        desciption: "Demo desciption about kitten 3"
    }
]
const clickMe = () => {
    alert("Thanks for clicking me. Hope you have a nice day!")
}

const addCards = (items) => {
    items.forEach(item => {
        let itemToAppend = '<div class="col s4 center-align">'+
    '<div class="card medium"><div class="card-image waves-effect waves-block waves-light"><img class="activator" src="'+item.image+'">'+
    '</div><div class="card-content">'+
    '<span class="card-title activator grey-text text-darken-4">'+item.title+'<i class="material-icons right">more_vert</i></span><p><a href="#">'+item.link+'</a></p></div>'+
    '<div class="card-reveal">'+
        '<span class="card-title grey-text text-darken-4">'+item.title+'<i class="material-icons right">close</i></span>'+
        '<p class="card-text">'+item.desciption+'</p>'+
    '</div></div></div>';
        $("#card-section").append(itemToAppend)
    });
}



$(document).ready(function(){
    $('.materialboxed').materialbox();
    $('#clickMeButton').click(()=>{
        clickMe();
    })
    addCards(cardList);
});
```

Advance Components Materialize Cont ...

Now lets run it and see how our application looks like.



Advance Components Materialize Cont ...

And that's how we add dynamic components to our webpage using a bit of jquery and materialize.

Now let's try to use another component which is used in all web applications is **Modal.**

I have attached a link where you can find how to add modals to your application and most of the time what we see is modals are used in order to open some forms on our web applications and that is what we would try to create. So let's get into it.

Advance Components Materialize Cont ...

So we update the index.html body and in the section main we make changes and make it look something like this.

Advance Components Materialize Cont ...

```
    <main class="main-body">
      <div class="container">
        <div class="row">
          <div class="col s12 center-align">
            <h1 id="heading">Welcome to SIT 725 Week 3</h1>
            <p class="flow-text">This is a demo application that tries to use a few components from materialize and through this practical we try to have some fun with Materialize and its
components</p>
          </div>
          <div class="col s12 center-align">
            <img class="materialboxed materialboxed-image" src="images/livedemo.png">
          </div>
          <div class="col s12 center-align">
            <a class="waves-effect waves-light btn click-me-button modal-trigger" id="clickMeButton" data-target="modal1"><i class="material-icons left">auto_fix_high</i>Click Me</a>
          </div>
        </div>
        <div class="row" id="card-section">
          <div class="col s4 center-align">
            <div class="card medium">
              <div class="card-image waves-effect waves-block waves-light">
                <img class="activator" src="images/kitten.jpg">
              </div>
              <div class="card-content">
                <span class="card-title activator grey-text text-darken-4">Kitten<i class="material-icons right">more_vert</i></span>
                <p><a href="#">About kitten</a></p>
              </div>
              <div class="card-reveal">
                <span class="card-title grey-text text-darken-4">Kitten<i class="material-icons right">close</i></span>
                <p class="card-text">Hello There! I just wanted to say HI to you guys. See ya!</p>
              </div>
            </div>
          </div>
        </div>
      </div>
        <!-- Modal Structure continues on next slide-->
```

Advance Components Materialize Cont ...

```
    <!-- Modal Structure -->
  <div id="modal1" class="modal modal-fixed-footer">
    <div class="row modal-content">
      <form class="col s12">
        <div class="row">
          <div class="input-field col s6">
            <input placeholder="Placeholder" id="first_name" type="text" class="validate">
            <label for="first_name">First Name</label>
          </div>
          <div class="input-field col s6">
            <input id="last_name" type="text" class="validate">
            <label for="last_name">Last Name</label>
          </div>
        </div>
        <div class="row">
          <div class="input-field col s12">
            <input id="password" type="password" class="validate">
            <label for="password">Password</label>
          </div>
        </div>
        <div class="row">
          <div class="input-field col s12">
            <input id="email" type="email" class="validate">
            <label for="email">Email</label>
          </div>
        </div>
        <div class="row">
          <div class="input-field col s12 center-align">
            <a class="waves-effect waves-light btn" id="formSubmit">submit</a>
          </div>
        </div>
      </form>
    </div>
  </div>
</main>
```

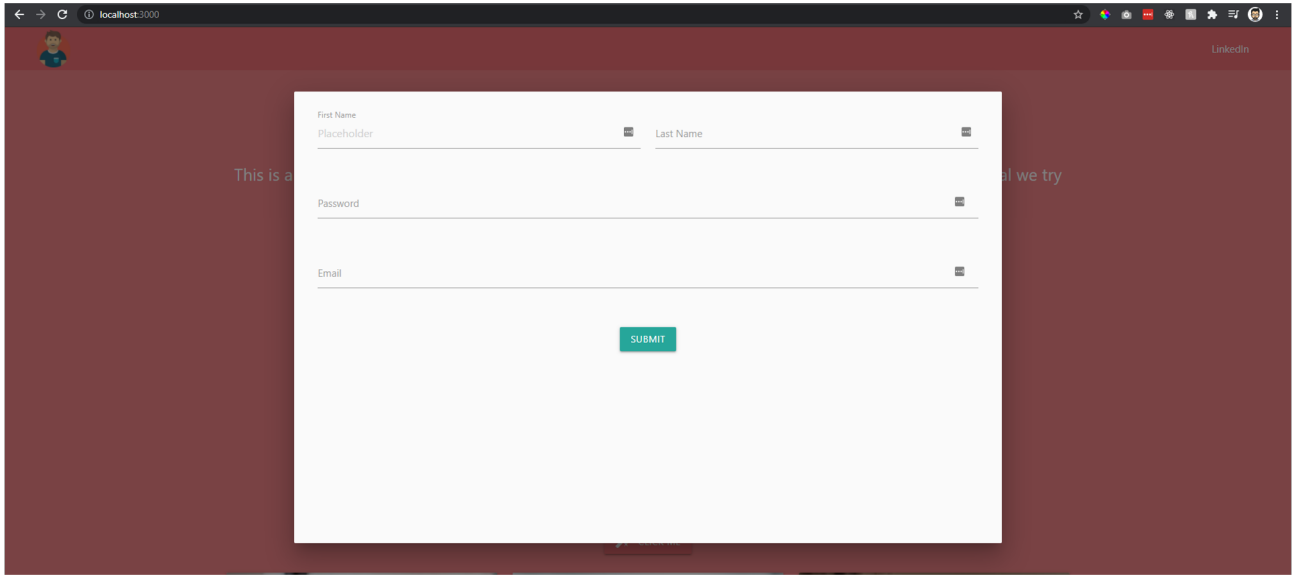
Advance Components Materialize Cont ...

We also update the script.js file and add some new functionalities to it.

```
const cardList = [
    {
        title: "Kitten 2",
        image: "images/kitten-2.jpg",
        link: "About Kitten 2",
        description: "Demo description about kitten 2"
    },
    {
        title: "Kitten 3",
        image: "images/kitten-3.jpg",
        link: "About Kitten 3",
        description: "Demo description about kitten 3"
    }
]
const clickMe = () => {
    alert("Thanks for clicking me. Hope you have a nice day!")
}

const submitForm = () => {
    let formData = {};
    formData.first_name = $('#first_name').val();
    formData.last_name = $('#last_name').val();
    formData.password = $('#password').val();
    formData.email = $('#email').val();

    console.log("Form Data Submitted: ", formData);
}

const addCards = (items) => {
    items.forEach(item => {
        let itemToAppend = '<div class="col s4 center-align">'+
    '<div class="card medium"><div class="card-image waves-effect waves-block waves-light"><img class="activator" src="'+item.image+'">'+
    '</div><div class="card-content">'+
    '<span class="card-title activator grey-text text-darken-4">'+item.title+'<i class="material-icons right">more_vert</i></span><p><a href="#">'+item.link+'</a></p></div>'+
    '<div class="card-reveal">'+
      '<span class="card-title grey-text text-darken-4">'+item.title+'<i class="material-icons right">close</i></span>'+
      '<p class="card-text">'+item.description+'</p>'+
    '</div></div></div>';
      $("#card-section").append(itemToAppend)
    });
}


$(document).ready(function(){
    $('.materialboxed').materialbox();
    $('#formSubmit').click(()=>{
        submitForm();
    })
    addCards(cardList);
    $('.modal').modal();
  });
```

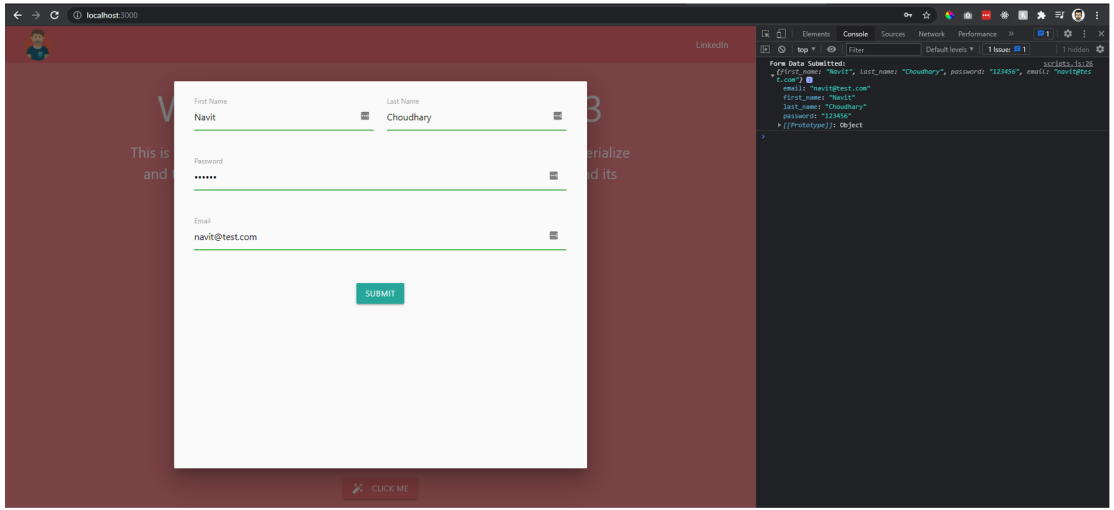
Advance Components Materialize Cont ...

And now that we run application and try to click the button Click Me we can see the modal opening.



Advance Components Materialize Cont ...

And when we fill up the form and try to submit the button you can open up the chrome debug console by just right clicking and pressing inspect and then go to the console tab, and now when you click the submit button you can see the data that you added to the form getting printed in the console.



Advance Components Materialize Cont ...

And using that data you can simply send the data to another endpoint and and save the data in your database.

So now would be the perfect time to commit your code again so that we have everything updated in our github repo.

Conclusion

So this was a very simple practical where we learned about materialize and a few of it components.

Thanks