IoT Experimental Learning
Week 7 Journal

1. **Describe the experience and what you hope to gain from participating in the experience.**
    - This week's assignment was a very easy for me, I found no issues while trying to complete assignment.
        1. Reviewed weekly assignment material.
        2. Completed Coding assignment

2. **Provide an overview of tasks and key activities (training, discussions, labs, assessments, etc.) in which you were engaged during the week.**

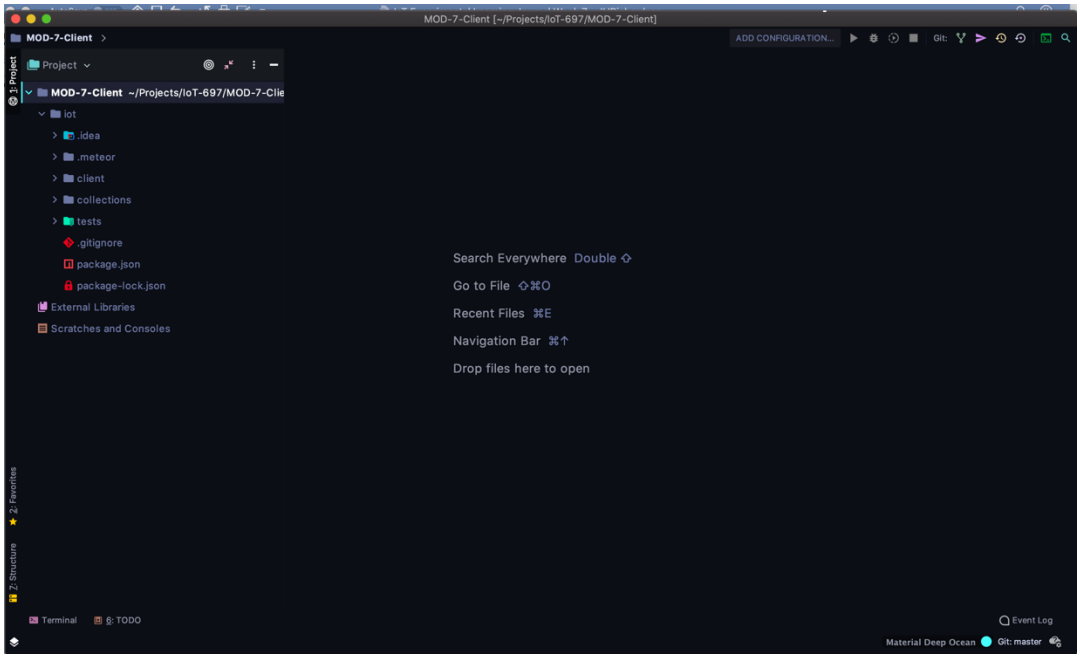
   For week 6 I accomplished the following tasks in chronological order;

❖ **Friday February 21, 2020 and Saturday February 22**, completed reviewing this week's assignment for the project.

- I reviewed the Module 7 weekly assignment on the SNHU Brightspace. https://learn.snhu.edu/d2l/le/content/343560/Home

- Started the Module 7 Coding assignment at:
    - https://learn.snhu.edu/content/enforced/343560-IT-697-X3325-OL-TRAD-GR.20TW3/Project%20Seven.pdf

- Created new Meteor Client "iot"



Created the following code to create the dashboard for sensor data

dashboard.css

```
/*
* Base structure
*/
/* Move down content because we have a fixed navbar that is 50px tall
*/
body {
    padding-top:50px;
}
 /*
 * Global add-ons
 */
.sub-header {
    padding-bottom: 10px;
    border-bottom: 1px solid #eee;
}
/*
* Top navigation* Hide default border to remove 1px line.
*/
.navbar-fixed-top{
    border:0;
}

/*
* Sidebar
*/
/*
* Hide for mobile, show later
*/
.sidebar {
    display:none;
}
@media (min-width:768px)
{
    .sidebar {
        position:fixed;
        top:51px;
        bottom:0;
        left:0;
        z-index:1000;
        display:block;
        padding:20px;
        overflow-x:hidden;
        overflow-y:auto;
        /* Scrollable contents if viewport is shorter than content.
        */
        background-color:#f5f5f5;
        border-right:1px solid#eee;
    }
}
/* Sidebar navigation
 */
.nav-sidebar {
    margin-right:-21px;
    /*
    * 20px padding + 1px border
    */
    margin-bottom:20px;
```

```
    margin-left:-20px;
}
.nav-sidebar > li > a {
    padding-right:20px;
    padding-left:20px;
}

.nav-sidebar > .active > a,
.nav-sidebar > .active > a:hover,
.nav-sidebar > .active > a:focus {
    color:#fff;
    background-color:#428bca;
}
/*
* Main content
*/
.main {
    padding:20px;
}
@media (min-width:768px) {
    .main {
        padding-right:40px;
        padding-left:40px;
    }
}
.main .page-header {
    margin-top:0;
}
/*
* Placeholder dashboard ideas
*/
.placeholders {
    margin-bottom:30px;
    text-align:center;
}
.placeholders h4 {
    margin-bottom:0;
}
.placeholder {
    margin-bottom:20px;
}
.placeholder img {
    display: inline-block;
    border-radius:50%;
}
```

dashboard.html

```
<template name="dashboard">
    <h1 class="page-header">Dashboard</h1>
    {{> sensorTable}}
</template>
```

head.html

```
<head>
    <meta charset = "utf-8">
    <meta http-equiv="X-UA-Compatible"content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>IoT</title>
</head>
```

home.html

```
<template name="home">
    <h1 class="page-header">Home</h1>
    <h2>Welcome to the IoT Home Page!</h2>
</template>
```

main-layout.html

```
<template name="mainLayout">
    {{> topNav}}
    <div class="container-fluid">
        <div class="row">
            {{#if currentUser}}
            {{> sideNav}}
            <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
                {{> yield}}
            </div>
            {{/if}}
        </div>
    </div>
</template>
```

sensorTable.html

```
<template name="sensorTable">
    <div class="table-responsive">
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>Sensor ID</th>
                    <th>Timestamp</th>
                    <th>Date:Time</th>
                    <th>Temperature</th>
                    <th>Humidity</th>
                </tr>
            </thead>
            <tbody>
                {{#each sensors}}
                    {{> sensorRow}}
                {{/each}}
            </tbody>
        </table>
    </div>
</template>

<template name="sensorRow">
    <tr>
        <td>{{_id}}</td>
        <td>{{timestamp}}</td>
        <td>{{datetime}}</td>
        <td>{{data.temperature}}</td>
        <td>{{data.humidity}}</td>
    </tr>
</template>
```

sideNav.html

```
<template name="sideNav">
    <div class="col-sm-3 col-md-2 sidebar">
        <ul class="nav nav-sidebar">
            <li>
                <a href="home">Home</a>
```

```
            </li>
            <li>
                <a href="dashboard">Dashboard</a>
            </li>
        </ul>
    </div>
</template>
```

topNav.html

```
<template name="topNav">
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed"
                        data-toggle="collapse" data-target="#navbar"
                        aria-expanded="false" aria-controls="navbar">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">IoT</a>
            </div>
            <div id="navbar" class="navbar-collapse collapse">
                <ul class="col-sm-3 nav navbar-nav navbar-right">
                    <li>
                        <a>{{> loginButtons}}</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</template>
```

Created the following javascript files to manage the data and control program flow

router.js

```

Router.configure({
    layoutTemplate:'mainLayout'
});

// Default route
Router.route('/',function() {
    Router.go('home');
});

Router.route('/home',function() {
    this.render('home');
});

Router.route('/dashboard',function() {
    this.render('dashboard');
});
```

sensor-table.js

```
import  {Sensors} from "../collections/collections";
```

```
Template.sensorTable.helpers({
    sensors:function() {
        return Sensors.find();
    }});

Template.sensorRow.helpers({
    datetime:function() {
        return Date(Template.currentData().timestamp);
    }});
```

Collections.js

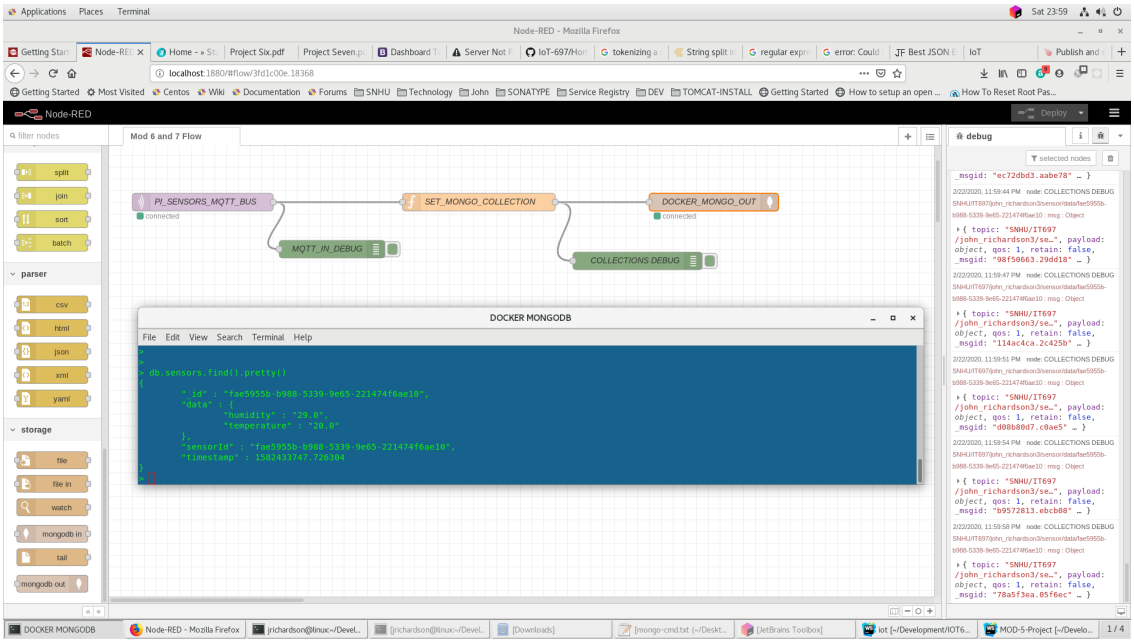
```
import { Meteor } from 'meteor/meteor';
import { Mongo } from 'meteor/mongo';


export const Sensors = new Mongo.Collection('sensors');

if (Meteor.isClient) {
    Meteor.subscribe('allSensors');
}

if(Meteor.isServer) {
        Meteor.publish('allSensors',function() {
            // only publish when logged-in users request publishing
            if(this.userId) {
                return[Sensors.find()];
            }
        });
    }
```

Updated package.json for application start and mongo server location as docker container.

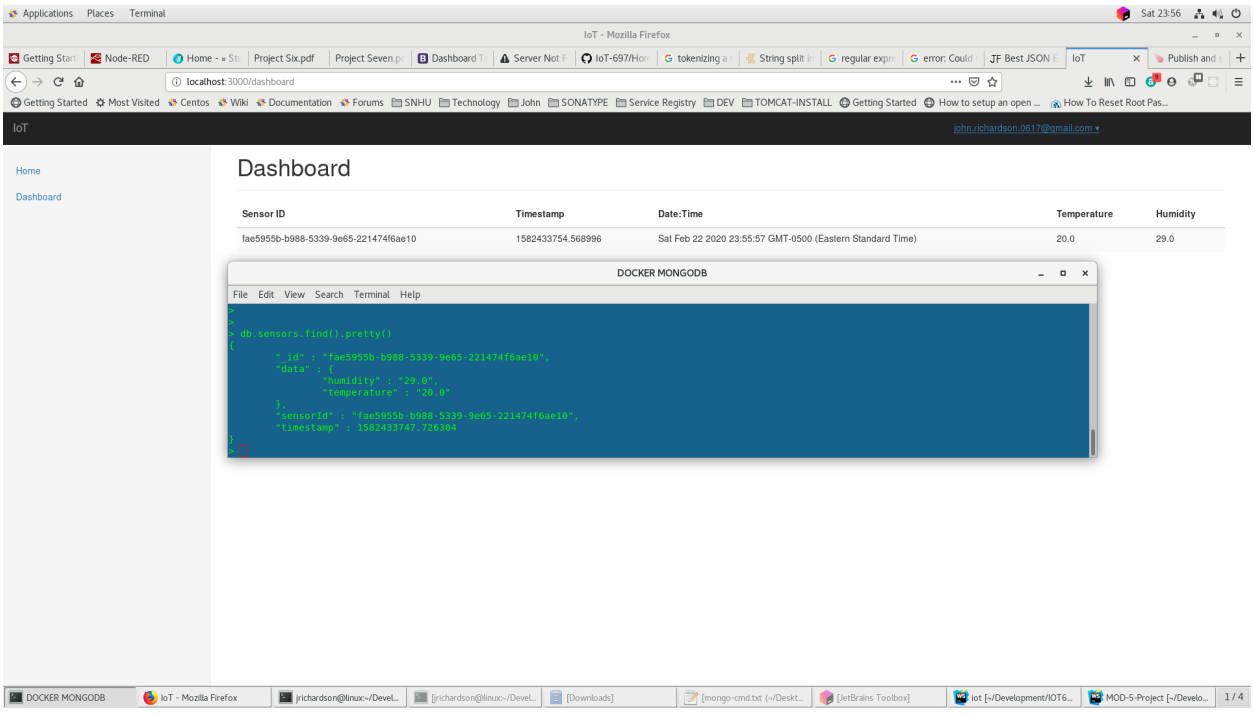
package.json

```
{
  "name": "iot",
  "private": true,
  "scripts": {
    "start": "MONGO_URL=mongodb://localhost:27017/meteor meteor --port 3002 run",
    "test": "meteor test --once --driver-package meteortesting:mocha",
    "test-app": "TEST_WATCH=1 meteor test --full-app --driver-package
meteortesting:mocha",
    "visualize": "meteor --production --extra-packages bundle-visualizer"
  },
  "dependencies": {
    "@babel/runtime": "^7.7.6",
    "bcrypt": "^4.0.0",
    "jquery": "^3.4.1",
    "meteor-node-stubs": "^1.0.0"
  }
}
```

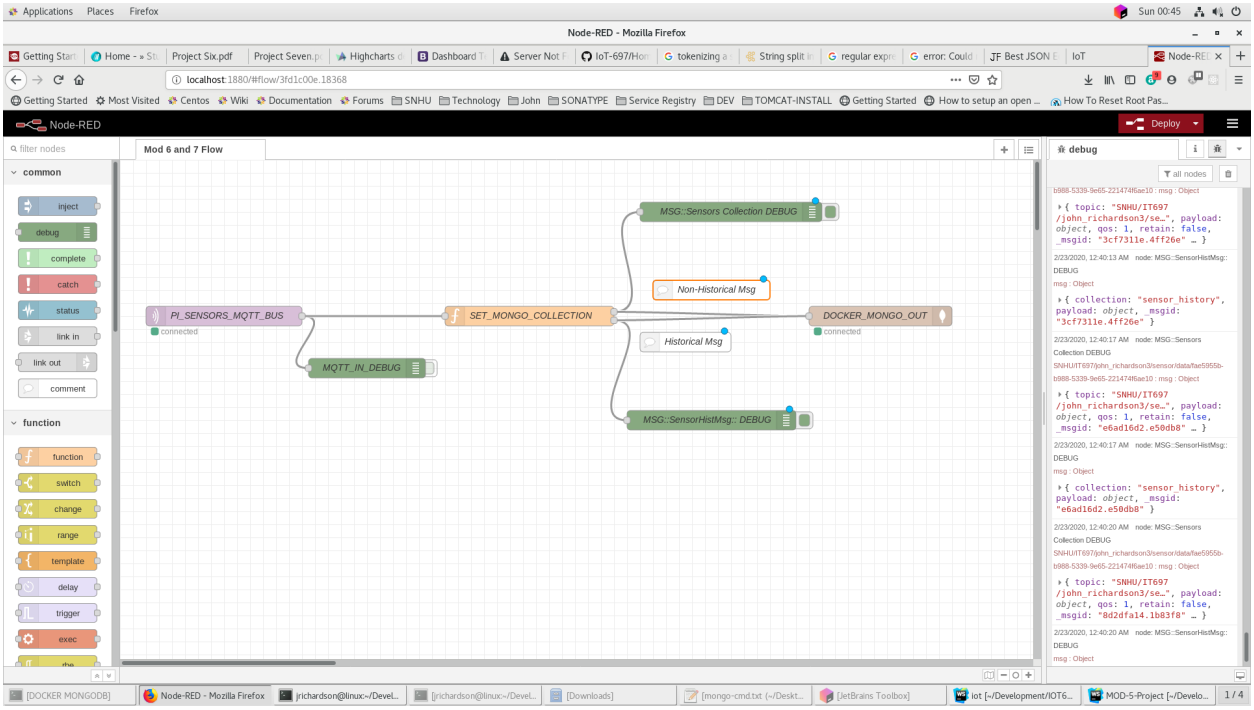
Work completed during Mod-6 and brought forward to Mod-7 as base



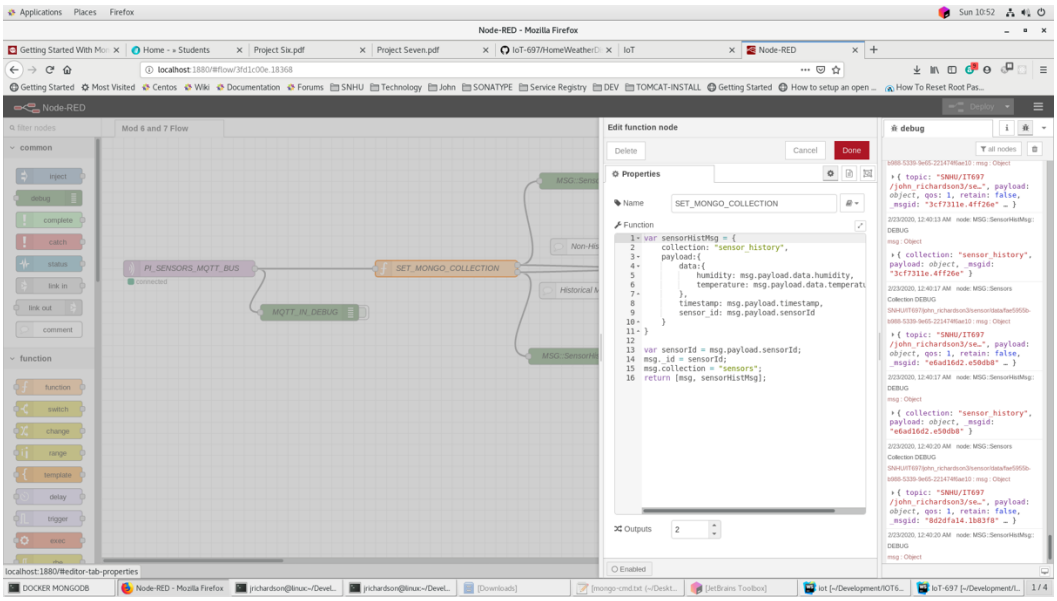
Running meteor client showing single updating temp-hum sensor with the same "_id"



Updated Flow to allow for 2 message outputs to support “Dashboard” and to provide sensor historical data to a new mongo collection “sensor_history” collection



A view of the javascript used to create the two messages



Observation: There were two methods in which to accomplish the above, I chose to complete this via the “node-red” flow as javascipt. After observing this running, I have determined that the best way to accomplish this is to utilize a table trigger in mongo to push only changes to the sensor_history collection. In its current state, there is many

redundant writes to the collection that should be over-looked until something has changed. I think I will change this in the future.