Name: Lakshmana Kumar Mettu

Class ID:11

Introduction:

- This ICP is get to know about D3(Data Driven Documents) creation and usage of data formats in given code and constructing a tree diagram in DOM model as well as creation of MongoDB account.
- Performing CRUD(create,Read,Update,Delete) operations on student deatils.

Programming Languages:

- HTML5
- Java script
- CSS

Software and platforms used:

- Jetbrains-webstorm
- Database in MongoDB

Objective1:

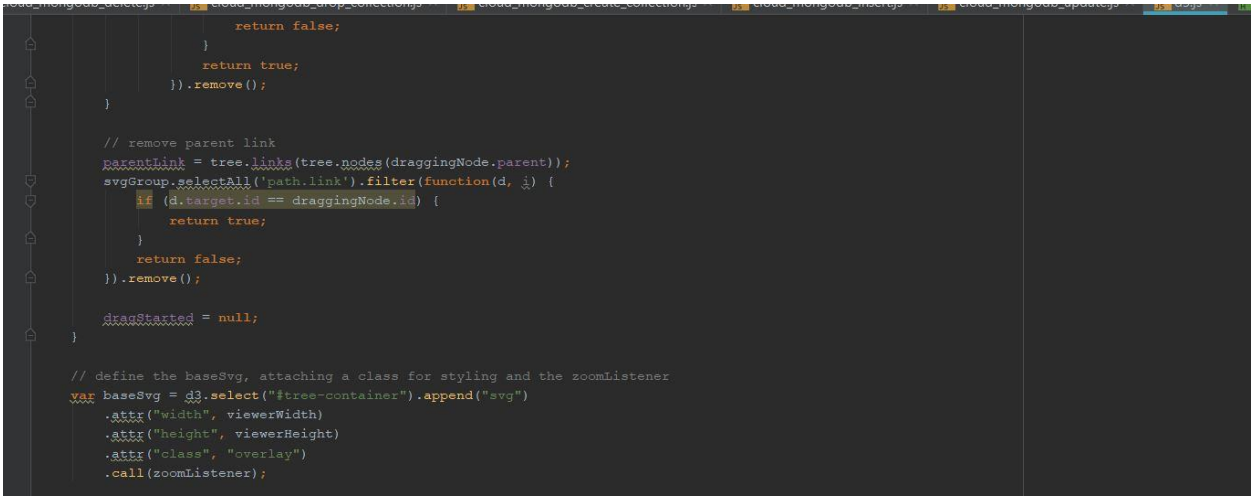
- For the given Use case D3 need to create of drag,drop,autosizing and collapse functions.
- For this D3 model data format as json is used.
- Functions for drag,drop,auto-sizing,collapse are created.
- The code snippets along with comments and output are given below.

```
    }
</style>
<script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="d3.js"></script>
<body>
<div id="tree-container"></div>
</body>
</html>
```

```
function sortTree() {
    tree.sort(function(a, b) {
        return b.name.toLowerCase() < a.name.toLowerCase() ? 1 : -1;
    });
}
// Sort the tree initially incase the JSON isn't in a sorted order.
sortTree();

//TODO: Pan function, can be better implemented.

function pan(domNode, direction) {
    var speed = panSpeed;
    if (panTimer) {
        clearTimeout(panTimer);
        translateCoords = d3.transform(svgGroup.attr("transform"));
        if (direction == 'left' || direction == 'right') {
            translateX = direction == 'left' ? translateCoords.translate[0] + speed : translateCoords.translate[0] - speed;
            translateY = translateCoords.translate[1];
        } else if (direction == 'up' || direction == 'down') {
            translateX = translateCoords.translate[0];
            translateY = direction == 'up' ? translateCoords.translate[1] + speed : translateCoords.translate[1] - speed;
        }
```

```
            }
            scaleX = translateCoords.scale[0];
            scaleY = translateCoords.scale[1];
            scale = zoomListener.scale();
            svgGroup.transition().attr("transform", "translate(" + translateX + "," + translateY + ")scale(" + scale + ")");
            d3.select(domNode).select('g.node').attr("transform", "translate(" + translateX + "," + translateY + ")");
            zoomListener.scale(zoomListener.scale());
            zoomListener.translate([translateX, translateY]);
            panTimer = setTimeout( handler: function() {
                pan(domNode, speed, direction);
            }, timeout: 50);
        }
    }

    // Define the zoom function for the zoomable tree

    function zoom() {
        svgGroup.attr("transform", "translate(" + d3.event.translate + ")scale(" + d3.event.scale + ")");
    }
```

```
    function initiateDrag(d, domNode) {
        draggingNode = d;
        d3.select(domNode).select('.ghostCircle').attr('pointer-events', 'none');
        d3.selectAll('.ghostCircle').attr('class', 'ghostCircle show');
        d3.select(domNode).attr('class', 'node activeDrag');

        svgGroup.selectAll("g.node").sort(function(a, b) { // select the parent and sort the path's
            if (a.id != draggingNode.id) return 1; // a is not the hovered element, send "a" to the back
            else return -1; // a is the hovered element, bring "a" to the front
        });
        // if nodes has children, remove the links and nodes
        if (nodes.length > 1) {
            // remove link paths
            links = tree.links(nodes);
            nodePaths = svgGroup.selectAll("path.link")
                .data(links, function(d) {
                    return d.target.id;
                }).remove();
            // remove child nodes
            nodesExit = svgGroup.selectAll("g.node")
                .data(nodes, function(d) {
                    return d.id;
                }).filter(function(d, i) {
                    if (d.id == draggingNode.id) {
```

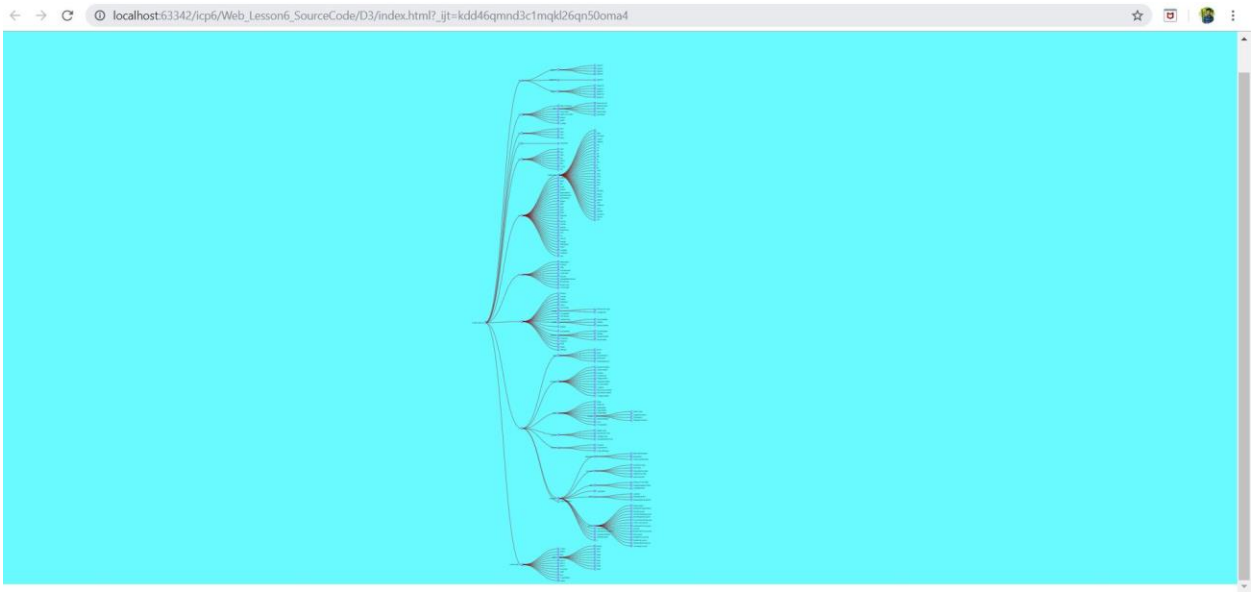


```
                    return false;
                }
                return true;
            }).remove();
        }

        // remove parent link
        parentLink = tree.links(tree.nodes(draggingNode.parent));
        svgGroup.selectAll('path.link').filter(function(d, i) {
            if (d.target.id == draggingNode.id) {
                return true;
            }
            return false;
        }).remove();

        dragStarted = null;
    }

    // define the baseSvg, attaching a class for styling and the zoomListener
    var baseSvg = d3.select("#tree-container").append("svg")
        .attr("width", viewerWidth)
        .attr("height", viewerHeight)
        .attr("class", "overlay")
        .call(zoomListener);
```

```
    // Define the drag listeners for drag/drop behaviour of nodes.
    dragListener = d3.behavior.drag()
        .on("dragstart", function(d) {
            if (d == root) {
                return;
            }
            dragStarted = true;
            nodes = tree.nodes(d);
            d3.event.sourceEvent.stopPropagation();
            // it's important that we suppress the mouseover event on the node being dragged. Otherwise it will absorb the mouseover event and the underlying no
        })
        .on("drag", function(d) {
            if (d == root) {
                return;
            }
            if (dragStarted) {
                domNode = this;
                initiateDrag(d, domNode);
            }

            // get coords of mouseEvent relative to svg container to allow for panning
            relCoords = d3.mouse($('svg').get(0));
            if (relCoords[0] < panBoundary) {
                panTimer = true;
```

```
// Helper functions for collapsing and expanding nodes.

function collapse(d) {
    if (d.children) {
        d._children = d.children;
        d._children.forEach(collapse);
        d.children = null;
    }
}

function expand(d) {
    if (d._children) {
        d.children = d._children;
        d.children.forEach(expand);
        d._children = null;
    }
}
```

```
"name": "words with A",
"children": [{
  "name": "A",
  "children": [{
    "name": "Apple0",
    "children": [{
      "name": "Apple1",
      "size": 3938
    }, {
      "name": "Apple2",
      "size": 3812
    }, {
      "name": "Apple3",
      "size": 6714
    }, {
      "name": "Apple4",
      "size": 743
    }]
  }, {
    "name": "Apple5",
    "children": [{
      "name": "Apple10",
      "size": 3534
    }, {
      "name": "apple12",
```

Objective-2:

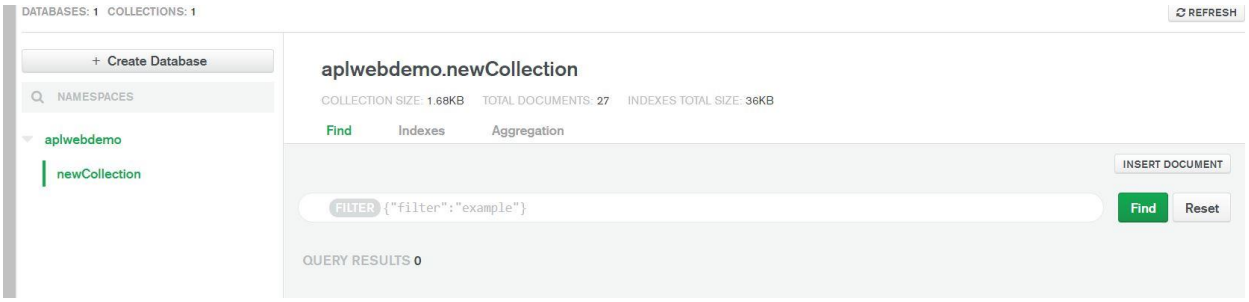
- To Create MongoDB account and create student database to perform CRUD operations.
- The code snippets along with output for each operation is given below.

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc0120@cluster0-iqytn.mongodb.net/test?retryWrites=true&w=majority'; //mongodb://<dbuser>:<dbpassword>@ds239128.mlab.c

MongoClient.connect(url, options: function(err, db) {
    if (err) throw err;
    console.log("Connected correctly to server");
    db.close();
```

```
"C:\Program Files\nodejs\node.exe" "C:\Users\laksh\WebstormProjects\web Technologies\icp6\Web_Lesson6_SourceCode\MongoDB\cloud_mongodb_create_collection.js"
Collection created!

Process finished with exit code 0
```

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc012@cluster0-iqytn.mongodb.net/test?retryWrites=true&w=majority';

MongoClient.connect(url, options: function(err, db) {
    if (err) throw err;
    var dbase = db.db("aplwebdemo");
    dbase.createCollection( name: "newCollection", options: function(err, res) {
        if (err) throw err;
        console.log("Collection created!");
        db.close();
    });
});
```

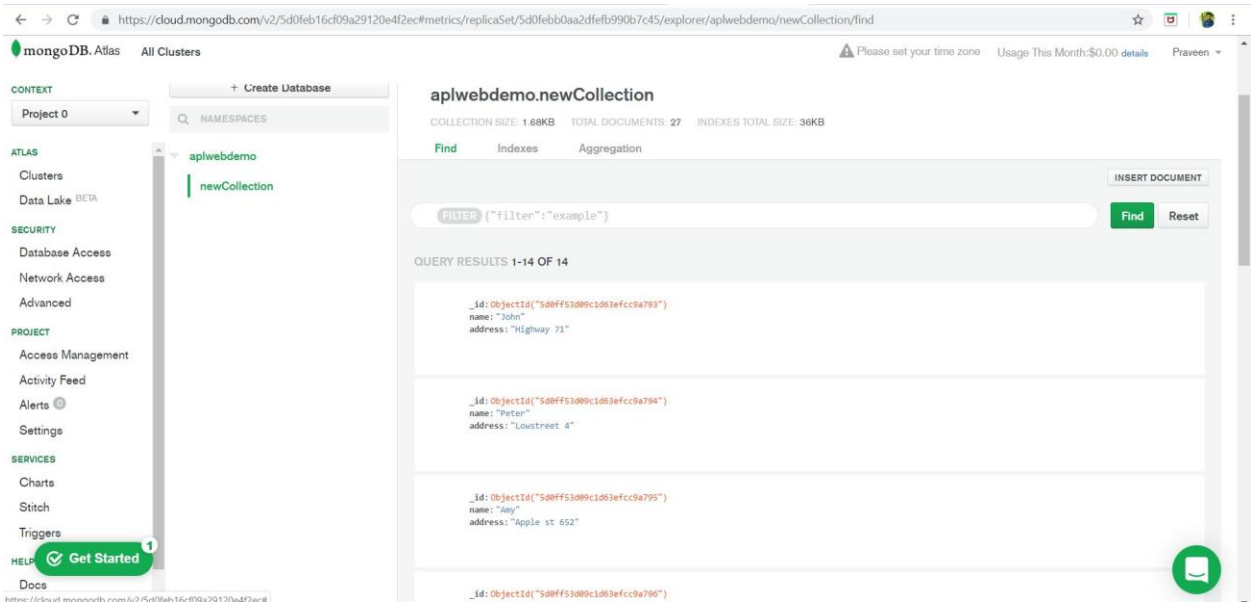
- Database Created and shown in snippet.



- Inserting Records In database.

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc012@cluster0-iqytn.mongodb.net/test?retryWrites=true&w=majority';

MongoClient.connect(url, options: function(err, db) {
    if (err) throw err;
    var dbase = db.db("aplwebdemo");
    var myobj = [
        { name: 'John', address: 'Highway 71'},
        { name: 'Peter', address: 'Lowstreet 4'},
        { name: 'Amy', address: 'Apple st 652'},
        { name: 'Hannah', address: 'Mountain 21'},
        { name: 'Michael', address: 'Valley 345'},
        { name: 'Sandy', address: 'Ocean blvd 2'},
        { name: 'Betty', address: 'Green Grass 1'},
        { name: 'Richard', address: 'Sky st 331'},
        { name: 'Susan', address: 'One way 98'},
        { name: 'Vicky', address: 'Yellow Garden 2'},
        { name: 'Ben', address: 'Park Lane 38'},
        { name: 'William', address: 'Central st 954'},
        { name: 'Chuck', address: 'Main Road 989'},
        { name: 'Viola', address: 'Sideway 1633'}
    ];
    dbase.collection("newCollection").insertMany(myobj, options: function(err, res) {
```

- Output Snippet for record insertion

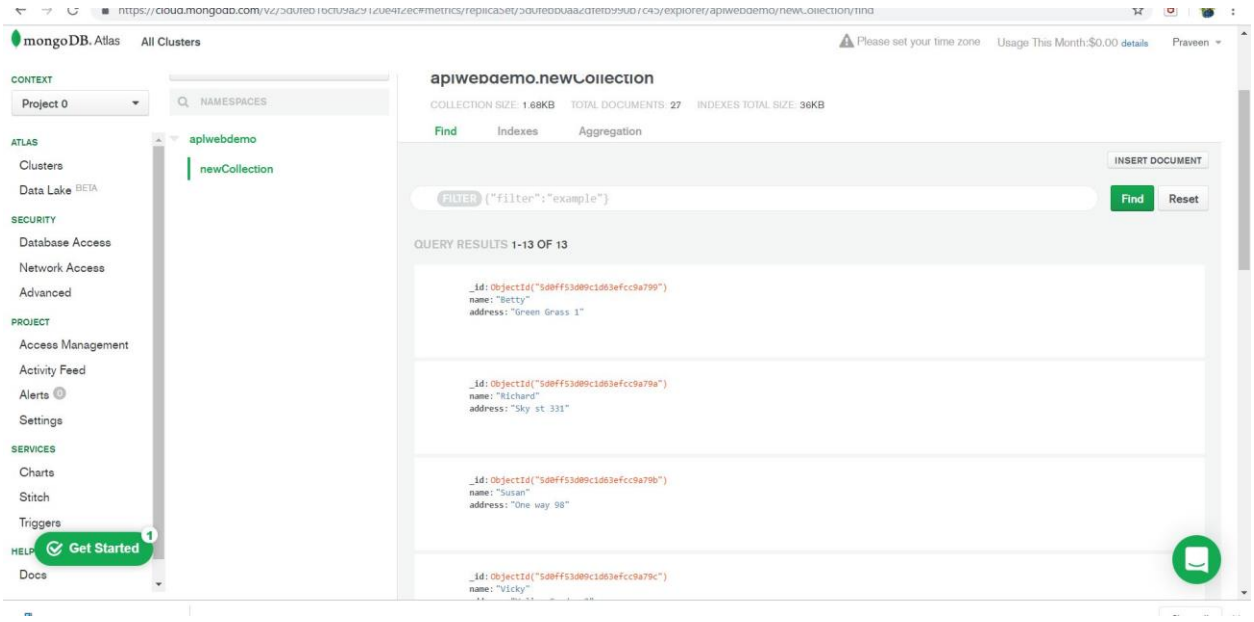


- Updating the Records

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc012@cluster0-iqytn.mongodb.net/test?retryWrites=false&w=majority';

MongoClient.connect(url, options: function(err, db) {
  if (err) throw err;
    var dbase = db.db("aplwebdemo");
    var myquery = { address: /^S/ };
    var newvalues = {$set: {name: "Minnie"} };
    var myoptions = { multi: true };
    dbase.collection("newCollection").updateMany(myquery, newvalues, myoptions, callback: function(err, res) {
        if (err) throw err;
        console.log(res.result.nModified + " record(s) updated");
        db.close();
    });
});
```

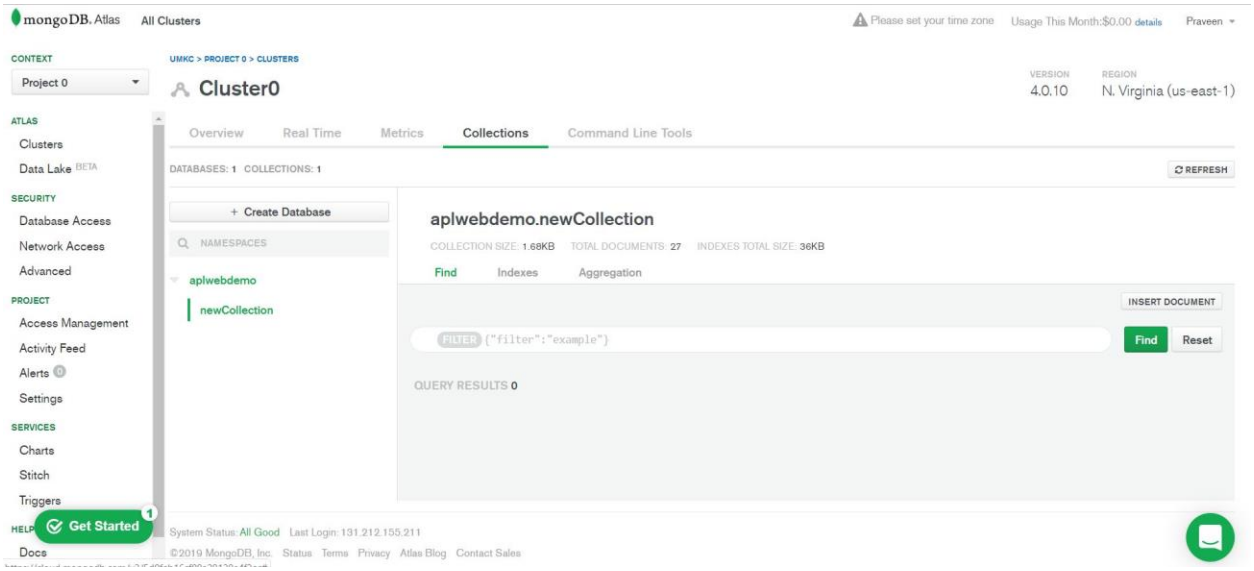
- Deleting 1 record

```
var http = require('http');
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc012@cluster0-iqytn.mongodb.net/test?retryWrites=true&w=majority';

MongoClient.connect(url, options: function(err, db) {
    if (err) throw err;
    var dbase = db.db("aplwebdemo");
    var myquery = { address: 'Main Road 989' };
    dbase.collection("newCollection").deleteOne(myquery, options: function(err, obj) {
        if (err) throw err;
        console.log(obj.result.n + " document(s) deleted");
        db.close();
    });
});
```



- Dropping Records

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb+srv://lakshmankumarmettu:ABCabc012@cluster0-iqytn.mongodb.net/test?retryWrites=true&w=majority';

MongoClient.connect(url, options: function(err, db) {
    if (err) throw err;
    var dbase = db.db("aplwebdemo");
    dbase.dropCollection( name: "newCollection", options: function(err, delOK) {
        if (err) throw err;
        if (delOK) console.log("Collection deleted");
        db.close();
    });
});
```



Discussions:

- Got keen knowledge on different data formats using D3 and also performing CRUD operations and learnt different data inserting techniques to cloud.

Conclusion:

- Hence learnt how to create Document Object Model(DOM) using Data Driven Document((D3) and also learnt to perform CRUD operations using MongoDB.