Module 6:Building Node.js Applications using ES6

Demo Document

edureka!

Dashboard Application Using ES6 And EJS

DEMO Steps:

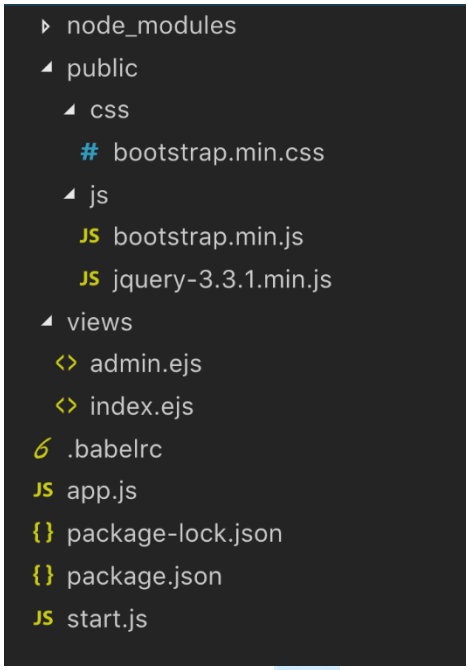
Step1: Create a package.json file.

```
Avyaans-MacBook-Pro:dashboard avi$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (dashboard) dashboard
version: (1.0.0)
description: Dashboard with Es6
entry point: (index.js)
test command:
git repository:
keywords: NodeJS Ejs
author: Edureka
license: (ISC)
About to write to /Users/avi/Desktop/folder/EdurekaApp/module6/dashboard/package.json:

{
  "name": "dashboard",
  "version": "1.0.0",
  "description": "Dashboard with Es6",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "NodeJS",
    "Ejs"
  ],
  "author": "Edureka",
  "license": "ISC"
}


Is this OK? (yes) yes
```

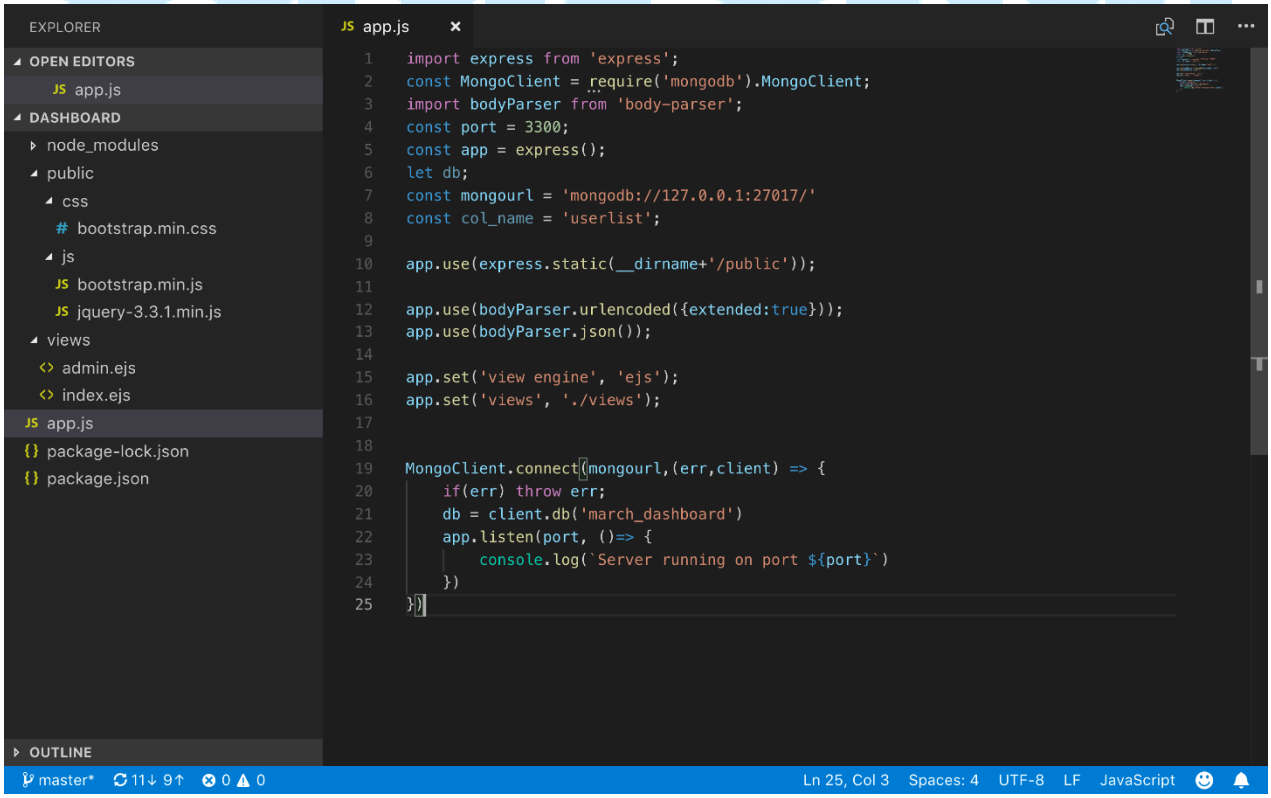
Step 2: Install express body-parse ejs and mongodb to create dashboard app.

```
Avyaans-MacBook-Pro:dashboard avi$ npm install express body-parser ejs mongodb
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN dashboard@1.0.0 No repository field.

+ ejs@2.6.1
+ mongodb@3.2.5
+ body-parser@1.19.0
+ express@4.17.0
added 60 packages from 45 contributors and audited 170 packages in 5.002s
found 0 vulnerabilities
```

Step 3: Create the folder structure as shown below. Here Public folder is for static content and View for EJS and rest is app code.



Step 4: Add connection string for mongodb, body parser as middleware and set EJS as view engine. At last we connect mongo with respective database(Do start the mongoDB server).



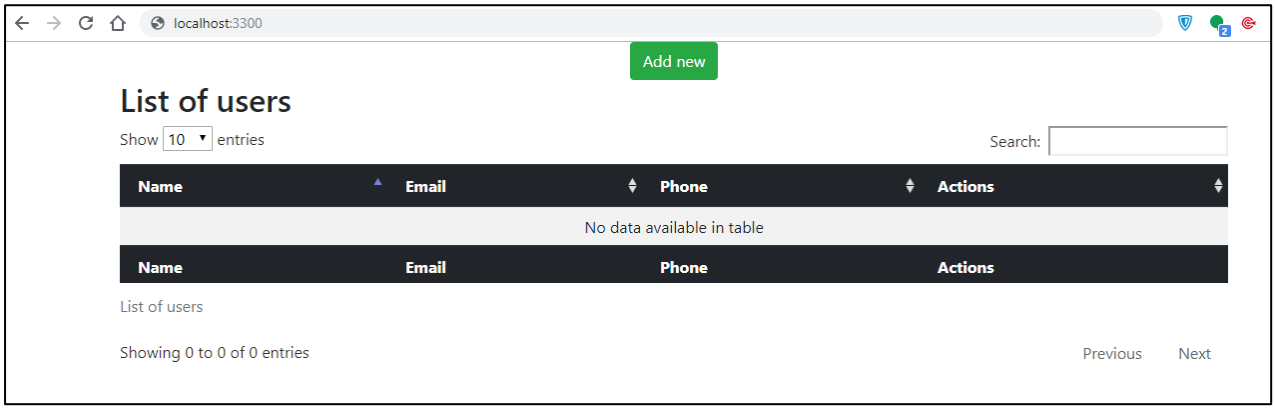
Step 5: Create a basic index.ejs file for view section that is connected with Css and bootstrap of local folder.

```
EXPLORER
OPEN EDITORS
  app.js
  index.ejs views
DASHBOARD
  node_modules
  public
    css
      bootstrap.min.css
    js
      bootstrap.min.js
      jquery-3.3.1.min.js
  views
    admin.ejs
    index.ejs
  app.js
  package-lock.json
  package.json

app.js   index.ejs ×

<html>
    <head>
        <title>Dashboard</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="../public/css/bootstrap.min.css">
        <script src="../public/js/bootstrap.min.js"></script>
        <script src="../public/js/bootstrap.min.js"></script>
        <script src="../public/js/jquery-3.3.1.min.js"></script>
        <!-- For DataTables -->
        <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.css">
        <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.js"></script>
    </head>
```

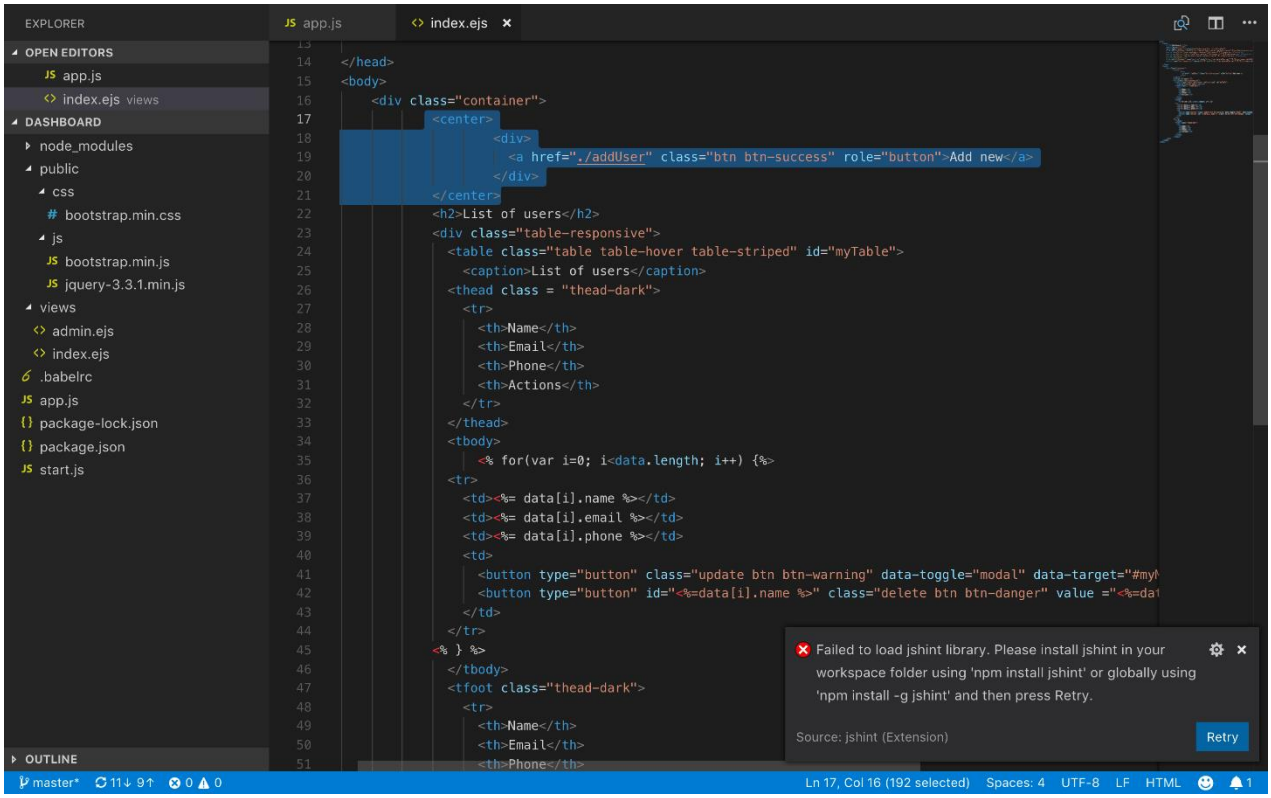
Step 6: Use Table of HTML and add script of datatable.js, use es6 syntax on line 29 to iterate over data.

```
EXPLORER
OPEN EDITORS
  app.js
  index.ejs views
DASHBOARD
  node_modules
  public
    css
      bootstrap.min.css
    js
      bootstrap.min.js
      jquery-3.3.1.min.js
  views
    admin.ejs
    index.ejs
  app.js
  package-lock.json
  package.json
OUTLINE

app.js   index.ejs ×

10  !-- For DataTables -->
11  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.css">
12  <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.js"></script>
13  ad>
14  y>
15  div class="container">
16          <h2>List of users</h2>
17          <div class="table-responsive">
18            <table class="table table-hover table-striped" id="myTable">
19              <caption>List of users</caption>
20            <thead class = "thead-dark">
21              <tr>
22                <th>Name</th>
23                <th>Email</th>
24                <th>Phone</th>
25                <th>Actions</th>
26              </tr>
27            </thead>
28            <tbody>
29                <% for(var i=0; i<data.length; i++) {%>
30            <tr>
31              <td><%= data[i].name %></td>
32              <td><%= data[i].email %></td>
33              <td><%= data[i].phone %></td>
34              <td>
35                <button type="button" class="update btn btn-warning" data-toggle="modal" data-target="#myModal" id="<%=data[i].name %>">Edi
36                <button type="button" id="<%=data[i].name %>" class="delete btn btn-danger" value ="<%=data[i].name %>">delete</button>
37              </td>
38            </tr>
39          <% } %>
40            </tbody>
41            <tfoot class="thead-dark">
42              <tr>
43                <th>Name</th>
44                <th>Email</th>
45                <th>Phone</th>
46                <th>Actions</th>
47              </tr>
48            </tfoot>
49            </table>
50  </div>
51  dy>
52
53

master*  11↓ 9↑  0 0                Ln 15, Col 32  Spaces: 4  UTF-8  LF  HTML
```

Step 7: Test your app on Browser.



Step 8: In EJS file add one anchor tag to open form, in order to take inputs from user and also to add new user.



Step 9: Create a new file admin.ejs and add HTML form and post path. Pass the post route to submit the data. Submit data this time using form instead of postman.

```
EXPLORER
OPEN EDITORS
  app.js
  admin.ejs views
  index.ejs views
DASHBOARD
  node_modules
  public
  views
    admin.ejs
    index.ejs
  .babelrc
  app.js
  package-lock.json
  package.json
  start.js
OUTLINE
```

```
<body>

<div class="container">
    <br>
  <nav class="navbar navbar-expand-md bg-dark navbar-dark fixed-top">
  <!-- Brand -->
  <a class="navbar-brand" href="#">Dashboard</a>

  <!-- Toggler/collapsibe Button -->
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#collapsibleNavbar">
    <span class="navbar-toggler-icon"></span>
  </button>

  <!-- Navbar links -->
  <div class="collapse navbar-collapse" id="collapsibleNavbar">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="/">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/new">Add new</a>
      </li>
    </ul>
  </div>
</nav>
  <h2>Add new Data</h2>
  <form action="/addData" method="POST">
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" class="form-control" id="name" placeholder="Enter Name" name="name" required>
    </div>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="text" class="form-control" id="email" placeholder="Enter Email" name="email">
    </div>
    <div class="form-group">
      <label for="phone">Phone:</label>
      <input type="text" class="form-control" id="phone" placeholder="Enter Phone" name="phone">
    </div>
    <button type="submit" class="btn btn-primary">Add</button>
    <a href="/" class="btn btn-success" role="button">cancel</a>
  </form>
</div>

</body>
</html>
```

master* 11↓ 9↑ 0 0 Ln 50, Col 11 Spaces: 4 UTF-8 LF HTML

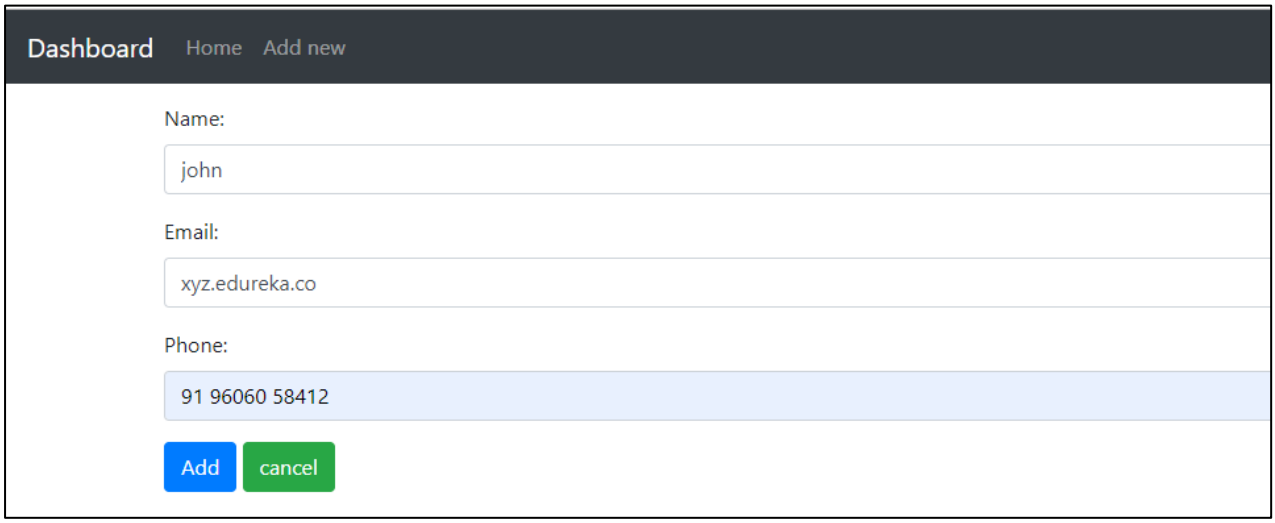
Step 10: Add a get route to render admin page using res.render and file name.

```
EXPLORER
OPEN EDITORS
  app.js
  index.ejs views
DASHBOARD
  node_modules
  public
    css
      bootstrap.min.css
    js
      bootstrap.min.js
      jquery-3.3.1.min.js
  views
    admin.ejs
    index.ejs
  .babelrc
  app.js
  package-lock.json
  package.json
  start.js
OUTLINE
```

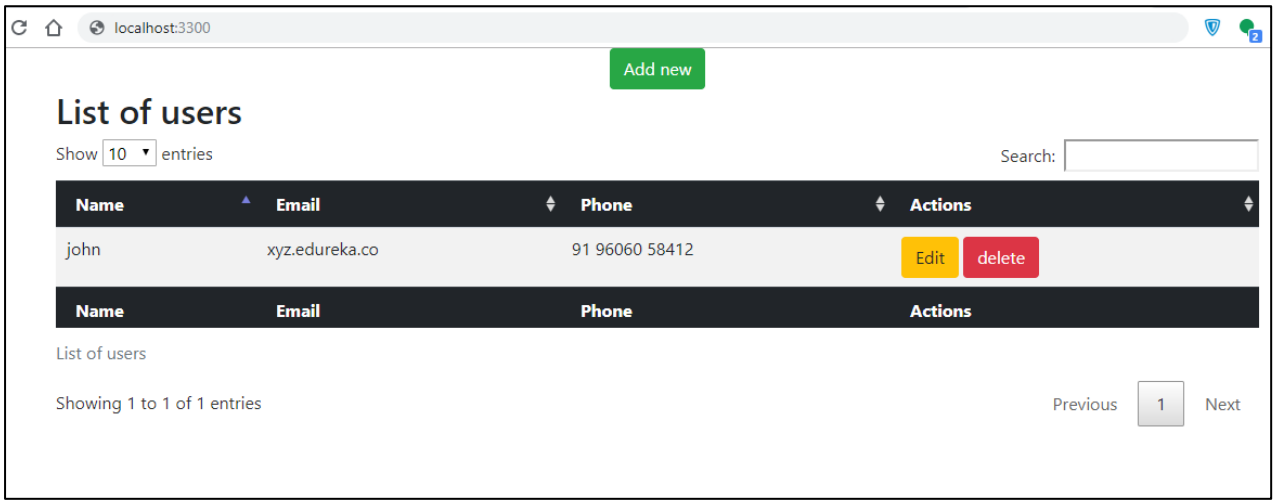
```
app.use(express.static(__dirname+'/public'));

app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());

app.set('view engine', 'ejs');
app.set('views', './views');

// Get Data from datbase and display on index
app.get('/', (req,res)=>{
    db.collection(col_name).find().toArray((err,result) => {
        if(err) throw err;
        var outupt = result;
        console.log(outupt)
        res.render('index.ejs',{data:result})
    })
})

// Opening Add User page
app.get('/addUser',(req,res) => {
    res.render('admin')
})

MongoClient.connect(mongourl,(err,client) => {
    if(err) throw err;
    db = client.db('march_dashboard')
    app.listen(port, ()=> {
        console.log(`Server running on port ${port}`)
    })
})
```

master* 11↓ 9↑ 0 0 Ln 28, Col 1 (85 selected) Spaces: 4 UTF-8 LF JavaScript

Step 11: Using form you can add new data into database and by clicking add you will send a post call to node app.



Step 12: You can see the user data in view list and this view list is fetched using Get api.



Step 13: Activate delete button by making call using Jquery Ajax and using id for each row.

```
EXPLORER
OPEN EDITORS
    app.js
    start.js
    admin.ejs views
    index.ejs views
DASHBOARD
    node_modules
    public
    views
        admin.ejs
        index.ejs
    .babelrc
    app.js
    package-lock.json
    package.json
    start.js
OUTLINE
```

```
27                  <tr>
28                    <th>Name</th>
29                    <th>Email</th>
30                    <th>Phone</th>
31                    <th>Actions</th>
32                  </tr>
33                </thead>
34                <tbody>
35                    <% for(var i=0; i<data.length; i++) {%>
36                <tr>
37                  <td><%= data[i].name %></td>
38                  <td><%= data[i].email %></td>
39                  <td><%= data[i].phone %></td>
40                  <td>
41                    <button type="button" class="update btn btn-warning" data-togg
42                      <button type="button" id="<%=data[i].name %>"
43                        class="delete btn btn-danger"
44                        value ="<%=data[i].name %>">
45                        delete
46                      </button>
47                  </td>
48                </tr>
49              <% } %>
50                </tbody>
51                <tfoot class="thead-dark">
52                  <tr>
53                    <th>Name</th>
54                    <th>Email</th>
55                    <th>Phone</th>
56                    <th>Actions</th>
57                  </tr>
```

master* 11↓ 9↑ 0 0 Ln 42, Col 25 (225 selected) Spaces: 4 UTF-8 LF HTML

Step 14: By activating delete button we need to delete route in Node.js app file.

```
EXPLORER
OPEN EDITORS
    app.js
    start.js
    admin.ejs views
    index.ejs views
    delete.js public/js
DASHBOARD
    node_modules
    public
        css
        js
            bootstrap.min.js
            delete.js
            jquery-3.3.1.min.js
    views
        admin.ejs
        index.ejs
    .babelrc
    app.js
    package-lock.json
    package.json
    start.js
OUTLINE
```

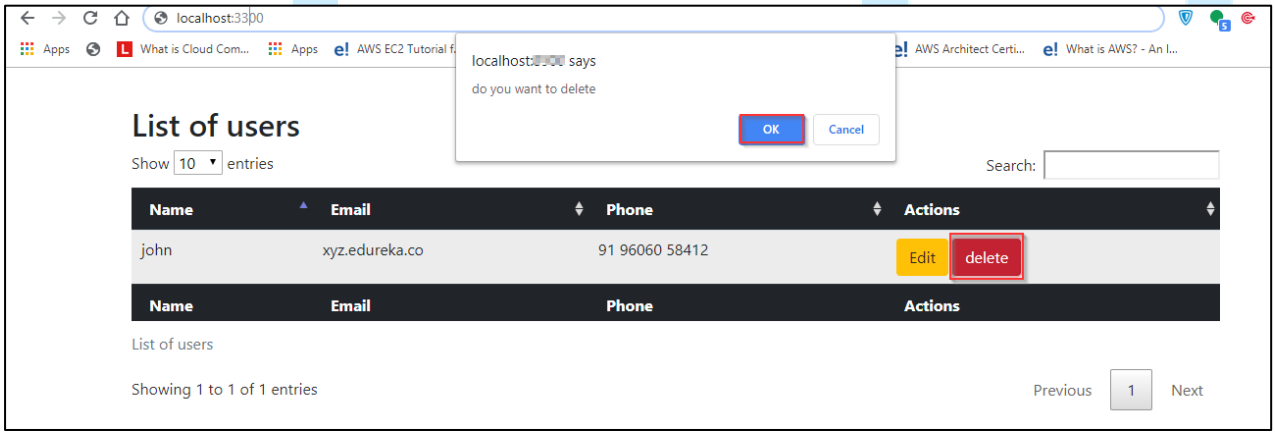
```
22          res.render('index.ejs',{data:result})
23      })
24  })
25
26  // Post data from ui
27  app.post('/addData', (req,res) => {
28      db.collection(col_name)
29          // In Req.body we will recive the data
30          // from form.
31          .insert(req.body, (err,result) => {
32              if(err) throw err;
33              console.log('data.inserted');
34          })
35      res.redirect('/');
36  })
37
38  // Delete Selected User
39  app.delete('/delete_user',(req,res) => {
40      db.collection(col_name).findOneAndDelete({
41          "name":req.body.name
42      },(err,result) => {
43          if (err) return res.send(500,err)
44          res.send({message: 'success'})
45      })
46  })
47
48  // Opening Add User page
49  app.get('/addUser',(req,res) => {
50      res.render('admin')
51  })
52
```

master* 11↓ 9↑ 0 0 Ln 38, Col 1 (255 selected) Spaces: 4 UTF-8 LF JavaScript

Step 15: Create a separate delete.js file to delete data (it is more like Ajax call running separately).

```
$('.delete').click(function() {
    var response = confirm("do you want to delete")
    name = this.id;
    console.log(response)
    if(response === true){
        $.ajax({
            type: 'DELETE',
            url: '/delete_user',
            method: 'delete',
            data: {"name":name},
            success: function(data){
                console.log('data is '+JSON.stringify(data));
                window.location.reload()
            },
            error: function(){
                alert('No data');
            }
        });
    }
    else{
        console.log("not deleted")
    }
});
```

Step 16: Check the working of delete button in Browser.



Step 17: Align edit button to update the request. On Click edit button you should be able to again call Ajax functions and pass id of each row record to be updated.

```
EXPLORER
OPEN EDITORS
  app.js
  start.js
  admin.ejs views
  index.ejs views
  delete.js public/js
DASHBOARD
  node_modules
  public
    css
    js
      bootstrap.min.js
      delete.js
      jquery-3.3.1.min.js
  views
    admin.ejs
    index.ejs
  .babelrc
  app.js
  package-lock.json
  package.json
  start.js
OUTLINE
```

```
34                  </thead>
35                  <tbody>
36                      <% for(var i=0; i<data.length; i++) {%>
37                  <tr>
38                    <td><%= data[i].name %></td>
39                    <td><%= data[i].email %></td>
40                    <td><%= data[i].phone %></td>
41                    <td>
42                      <button type="button" class="update btn btn-warning"
43                       data-toggle="modal" data-target="#myModal"
44                       id="<%=data[i].name %>">
45                          Edit
46                       </button>
47                        <button type="button" id="<%=data[i].name %>"
48                          class="delete btn btn-danger"
49                          value ="<%=data[i].name %>">
50                          delete
51                        </button>
52                    </td>
53                  </tr>
54                <% } %>
55                  </tbody>
56                  <tfoot class="thead-dark">
57                    <tr>
58                      <th>Name</th>
59                      <th>Email</th>
60                      <th>Phone</th>
61                      <th>Actions</th>
62                    </tr>
63                  </tfoot>
64                </table>
```

```
master*  11↓ 9↑  0 0        Ln 42, Col 22 (231 selected)  Spaces: 2  UTF-8  LF  HTML
```

Step 18: Create a bootstrap modal that popups with data on clicking edit button.

```
EXPLORER
OPEN EDITORS
  app.js
  start.js
  admin.ejs views
  index.ejs views
  delete.js public/js
DASHBOARD
  node_modules
  public
    css
    js
      bootstrap.min.js
      delete.js
      jquery-3.3.1.min.js
  views
    admin.ejs
    index.ejs
  .babelrc
  app.js
  package-lock.json
  package.json
  start.js
```

```
66                  <div class="modal fade" id="myModal">
67                    <div class="modal-dialog">
68                      <div class="modal-content">
69
70                        <!-- Modal Header -->
71                        <div class="modal-header">
72                          <h4 class="modal-title">Modal Heading</h4>
73                          <button type="button" class="close" data-dismiss="modal">&times;</button>
74                        </div>
75
76                        <!-- Modal body -->
77                        <div class="modal-body">
78                          <div class="container">
79                    <h2>Edit Data</h2>
80                    <form action="/update_user" method="POST" id="update_user">
81
82                      <div class="form-group">
83                        <label for="name">Name:</label>
84                        <input type="text" class="form-control" id="update_name" placeholder="Enter First name" name="name" requir
85                      </div>
86                      <div class="form-group">
87                        <label for="email">Email:</label>
88                        <input type="text" class="form-control" id="update_email" placeholder="Enter Email" name="email">
89                      </div>
90                      <div class="form-group">
91                        <label for="phone">Phone:</label>
92                        <input type="email" class="form-control" id="update_phone" placeholder="Enter phone" name="phone">
93                      </div>
94                    </form>
95                  </div>
96                        </div>
97
98                        <!-- Modal footer -->
99                        <div class="modal-footer">
100                         <button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>
101                         <button type="submit" id="update_table" class="btn btn-primary">update</button>
102                       </div>
103
104                   </div>
105                 </div>
106               </div>
107   script type="text/javascript" src="../js/delete.js"></script>
108   y>
109
```

Step 19: Created an update script to fetch data for each user populated in modal form (later on we can edit if required).

```
var name;
var to_be_updated;
        $(document).ready(function() {
            $('#myTable').DataTable();
            });
        // edit data
        $('.update').click(function() {
            id= this.id;
                $.ajax({
                    type: 'POST',
                    url: '/find_by_name',
                    data: {"name":id},
                    success: function(data){
                            to_be_updated = data[0].name;
                            $("#update_name").attr("value", data[0].name);
                            $("#update_email").attr("value", data[0].email);
                            $("#update_phone").attr("value", data[0].phone);
                            $('#Modal').modal({show: true});
                        },
                    error: function(){
                        alert('No data');
                    }
                    });
            });
```

Step 20: Create one post route on basis of user ID to fetch user data.

```
            console.log('data.inserted');
        })
    res.redirect('/');
})

// Delete Selected User
app.delete('/delete_user',(req,res) => {
    db.collection(col_name).findOneAndDelete({
        "name":req.body.name
    },(err,result) => {
        if (err) return res.send(500,err)
        res.send({message: 'success'})
    })
})

/ Find user by name
app.post('/find_by_name',(req,res) => {
    let name = req.body.name;
    db.collection(col_name)
      .find({name:name})
      .toArray((err,result) => {
          if(err) throw err;
          res.send(result)
      })
});

// Opening Add User page
app.get('/addUser',(req,res) => {
    res.render('admin')
})
```

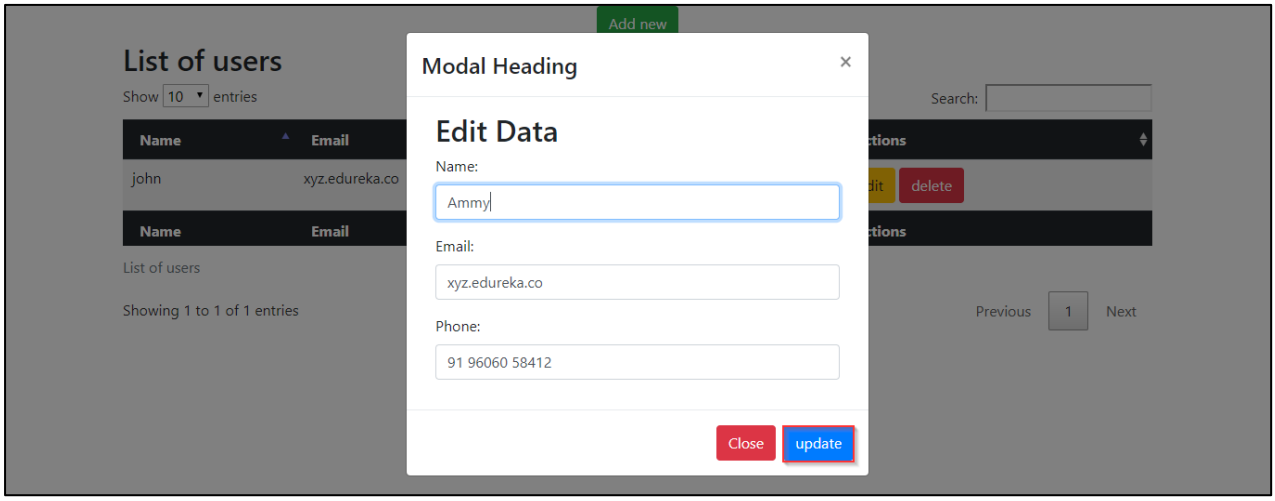
Step 21: Create one more Ajax call to update the record on clicking update button.

```
// update data
$(function(){
    $('#update_table').on('click', function(e){
      console.log('i am indsd');
      var data = $('#update_user').serialize();
      debugger;
      console.log(JSON.stringify(data));
      e.preventDefault();
      $.ajax({
        url: '/update_user',
        type:'PUT',
        data : data,
        success: function(data){
          console.log('i am googleapis');
          window.location.reload()
      },
      error: function(){
        alert('No data');
      }
    });
  });
  });
```

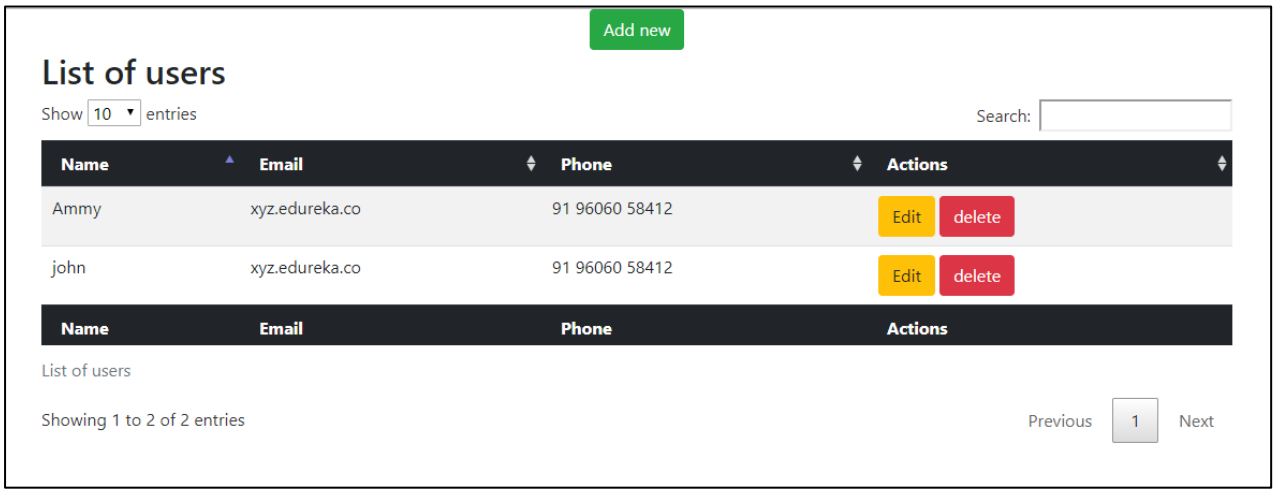
Step 22: To support step 20 initiate a PUT request to generate Update Query.

```
db.collection(col_name)
  .find({name:name})
  .toArray((err,result) => {
      if(err) throw err;
      res.send(result)
  })
});

// Update User
app.put('/update_user',(req,res)=>{
    db.collection(col_name)
        .findOneAndUpdate({"name":req.body.name},{
            $set:{
                name:req.body.name,
                email:req.body.email,
                phone:req.body.phone
            }
        },{
            upsert:true
        },(err,result) => {
            if(err) return res.send(err);
            res.send(result)
        })
})

// Opening Add User page
app.get('/addUser',(req,res) => {
    res.render('admin')
})

MongoClient.connect(mongourl,(err,client) => {
```

Step 23: Check the working of update button in Browser. When we click on edit, modal should popup with existing data.



Step 24: Finally, we can see updated data and dashboard CRUD operations complete



Conclusion:

We have successfully created Dashboard application using ES6 and EJS template