MEAN Stack Development

BUILDING MEAN STACK APPLICATION

CONTACT LIST CRUD APPLICATION

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY, CHANDIGARH

Submitted in Partial Fulfillment of the Requirement for the Certificate of MEAN STACK DEVELOPER Course 2020

(Course Code: ON025)

TO	***BY***
Sh. Sachin Chandla	**Srinivas Rahul Sapireddy**
Scientist 'D'/ Joint Director	**Roll No: 1**
NIELIT Chandigarh	**studUniqueCode: ONL/202006/Mea/000002**

ACKNOWLEDGEMNT

Foremost, I would like to express my sincere gratitude to my Course Coordinator Shri. Sachin Chandla who taught me MongoDB, Express and Node.JS. With his continuous support of my course study I understood the concepts of MEAN Stack Development and done project with his support. Sir made me to do project practically with his guidance.

Besides my advisor, I would like to thank Ms. Poonam Rana for encouragement, insightful comments and good sessions on Angular JS.

I once again thank Shri Sachin Chandla. His guidance helped me to complete writing my Project Report. I could not have imagined having a better Course Coordinator. He is very patience and a good motivator.

Rahul S Reddy

(SRINIVAS RAHUL SAPIREDDY)

CONTENTS

[1] What is MEAN Application?

MEAN in MEAN Stack Application represents the following acronyms:

M – Mongo DB

E – Express

A – Angular

N – Node.js

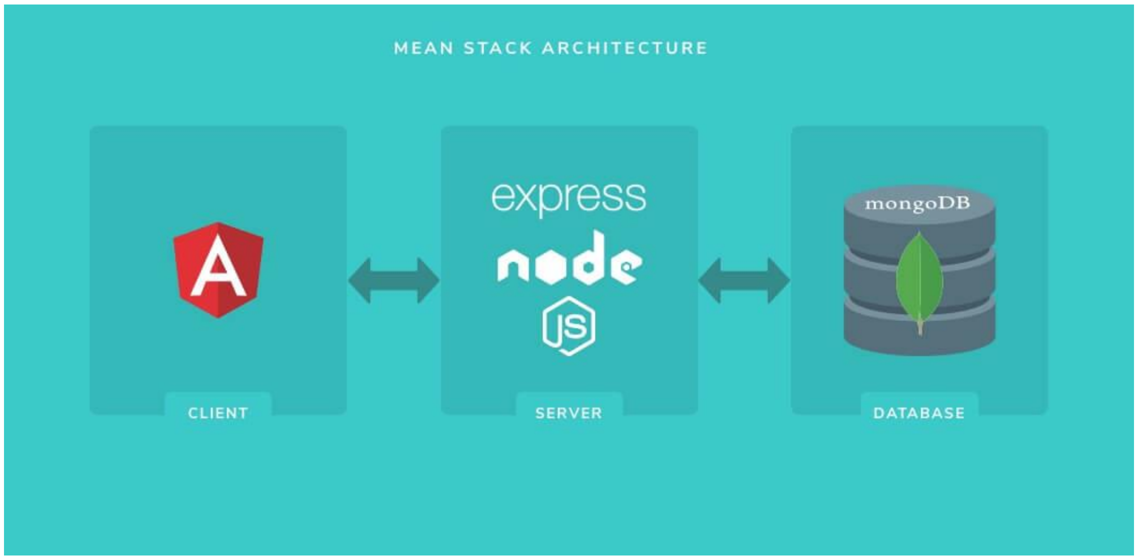
A collection of JavaScript technologies used for web development is called MEAN Stack. As shown in the figure from client side to the server side all the code is written in JavaScript.

Client Server Architecture of a MEAN Application

Client Side Technologies – Angular

Server Side Technologies – Express based on Node.js

Database – Mongo DB



MEAN Stack Architecture

(Picture Source: https://dzone.com/articles/mean-stack-amp-startups-are-they-made-for-each-oth)

Features of MEAN Stack Architecture

(a) Supports MVC (Model View Controller Architecture)
(b) Open Source
(c) Updated Regularly

[2] Mongo DB

(Picture Source: https://medium.com/plumbersofdatascience/mongo-db-basics-f1a2c74b2bfe)

Mongo DB is a non relational database system which uses a document-oriented data model. Here Mongo DB works on collections and documents.

Collection – Group of Document (Equivalent to a table in RDBMS System)

Document – Set of key value pairs

It is easy to pass data between client side and server side using Mongo DB as it saves data in the form of JSON.

Features of Mongo DB

(a) Fast Development
(b) Easy to store and retrieve data
(c) Strong Consistency

[3] Express

Express is a framework based on Node.js for creating web applications. We can build single page and multipage applications using express framework. Applications can be built in an easy manner using node framework which also used JavaScript.

(Picture Source: https://medium.com/javascript-in-plain-english/3-express-js-features-you-need-to-know-8f78b0035f33)

[4] Angular

(Picture Source: https://medium.com/@mnemon1ck/why-you-should-not-use-angularjs-1df5ddf6fc99)

Developed by Google. An open source popular framework used for building web applications. OOP is introduced into angular using typescript which made web development using angular more popular.

Here the client will be interacting with angular in the front end.

[5] Node.js

(Picture Source: https://coralogix.com/integrations/coralogix-nodejs-integration/)

An execution environment which resides in server side. Node.js is built on Google chrome v8 JavaScript engine.

It follows a single thread mechanism to process jobs (also called as events).

The event can be categorized as:

(a) Clicking a button
(b) Filling form etc

The processing of these events are done asynchronously which takes place in server and server in turn generates a response as an output.

In case of having many events coming in the node server the server processes multiple requests at the same time with no waiting time. Thus we can say that node architecture is a non blocking I/O which is single threaded taking less amount time for processing many events at a time.

Features of NodeJS

(a) Asynchronous Event Driven
(b) Highly Scalable
(c) Less Buffering Time

[6] CRUD Operation

CRUD Stands for:

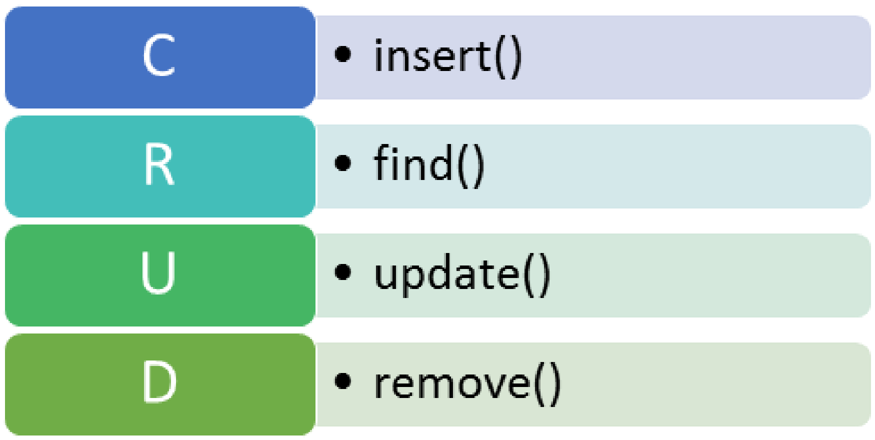
C - Create

R - Read

U - Update

D – Delete

These operations are performed on data present in Mongo DB.



(Picture Source: https://dzone.com/articles/getting-started-with-mongodb-part-2)

Methods associated with CURD Operations:

POST – To insert new data

GET – To retrieve data

PUT – For Modifying data

DELETE – For deleting data in database

All these operations are implemented in the project using RESTFul APIs.

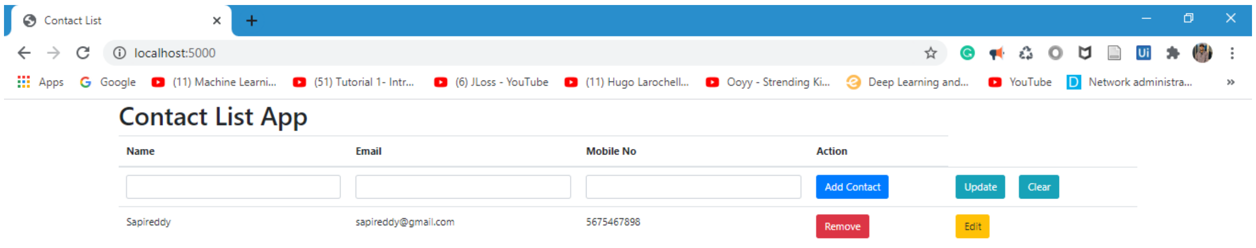
[7] Technology Requirements

UI:	Angular JS, HTML, CSS, JS, JSON
Platform:	Middleware & Server
Server Side:	Express, Node JS, REST API
Database:	Mongo DB
Text Editor:	Atom, Notepad ++
IDE:	Visual Studio Code

[8] Contact List MEAN Application (Running the Application with screenshots)

In this section we will see how to run MEAN Stack CRUD Application with screenshots.

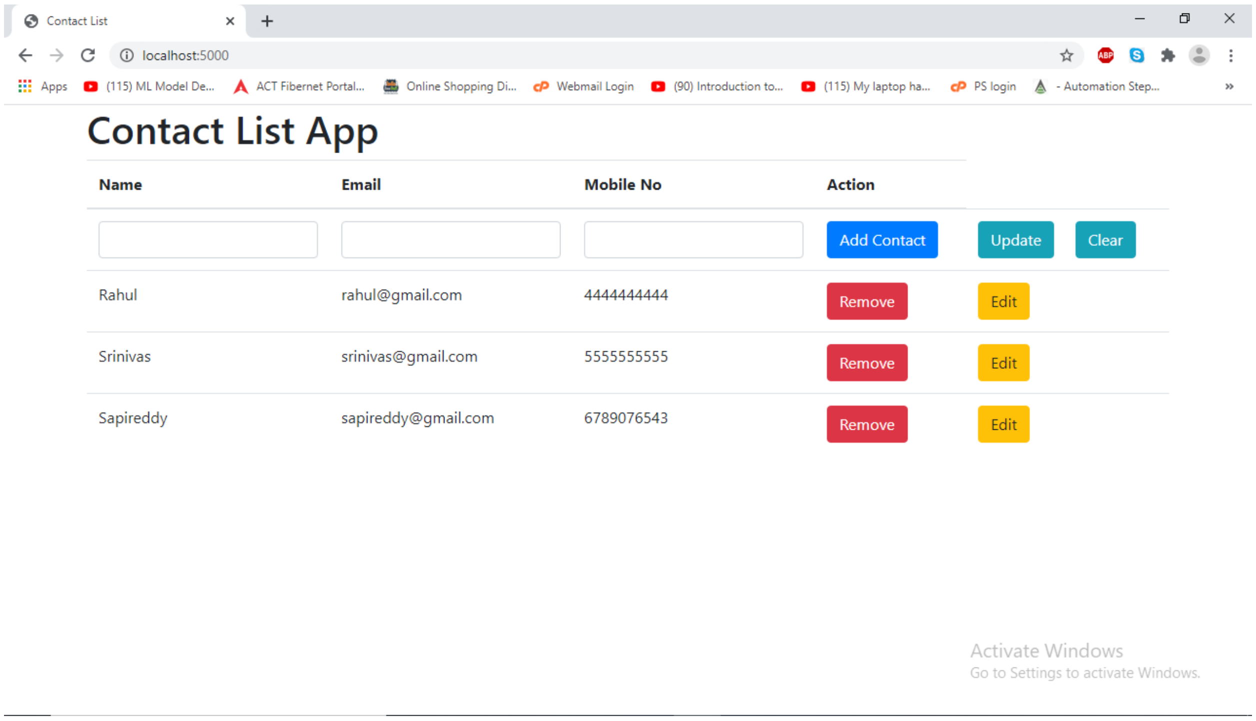
Running App **-** Type **node Server.js** to run application



Contact List App

Name	Email	Mobile No	Action		
			Add Contact	Update	Clear
Sapireddy	sapireddy@gmail.com	5675467898	Remove	Edit	

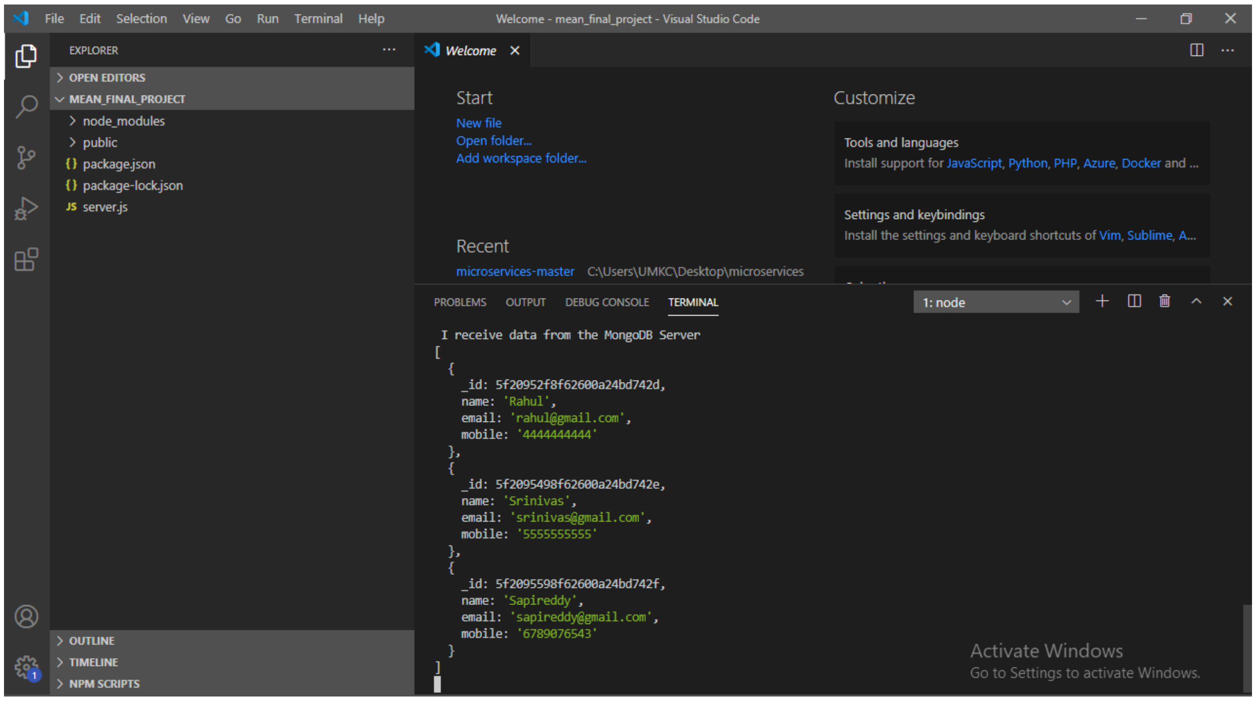
Application Running on PORT 5000



Contact List App

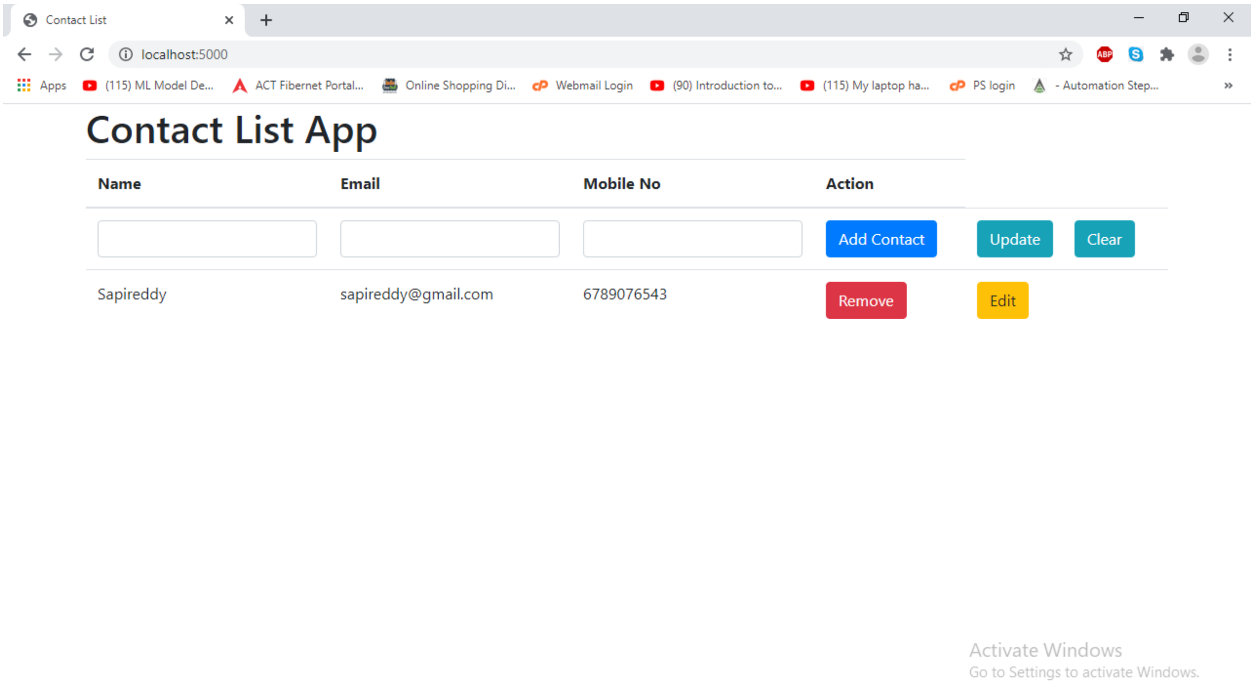
Name	Email	Mobile No	Action		
			Add Contact	Update	Clear
Rahul	rahul@gmail.com	4444444444	Remove	Edit	
Srinivas	srinivas@gmail.com	5555555555	Remove	Edit	
Sapireddy	sapireddy@gmail.com	6789076543	Remove	Edit	

Added 3 Contacts in the App


File Edit Selection View Go Run Terminal Help
Welcome - mean_final_project - Visual Studio Code
EXPLORER
OPEN EDITORS
MEAN_FINAL_PROJECT
node_modules
public
package.json
package-lock.json
server.js
OUTLINE
TIMELINE
NPM SCRIPTS
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
microservices-master C:\Users\UMKC\Desktop\microservices
Customize
Tools and languages
Install support for JavaScript, Python, PHP, Azure, Docker and ...
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, A...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: node
Activate Windows
Go to Settings to activate Windows.

```
 I receive data from the MongoDB Server
[
  {
    _id: 5f20952f8f62600a24bd742d,
    name: 'Rahul',
    email: 'rahul@gmail.com',
    mobile: '4444444444'
  },
  {
    _id: 5f2095498f62600a24bd742e,
    name: 'Srinivas',
    email: 'srinivas@gmail.com',
    mobile: '5555555555'
  },
  {
    _id: 5f2095598f62600a24bd742f,
    name: 'Sapireddy',
    email: 'sapireddy@gmail.com',
    mobile: '6789076543'
  }
]
```

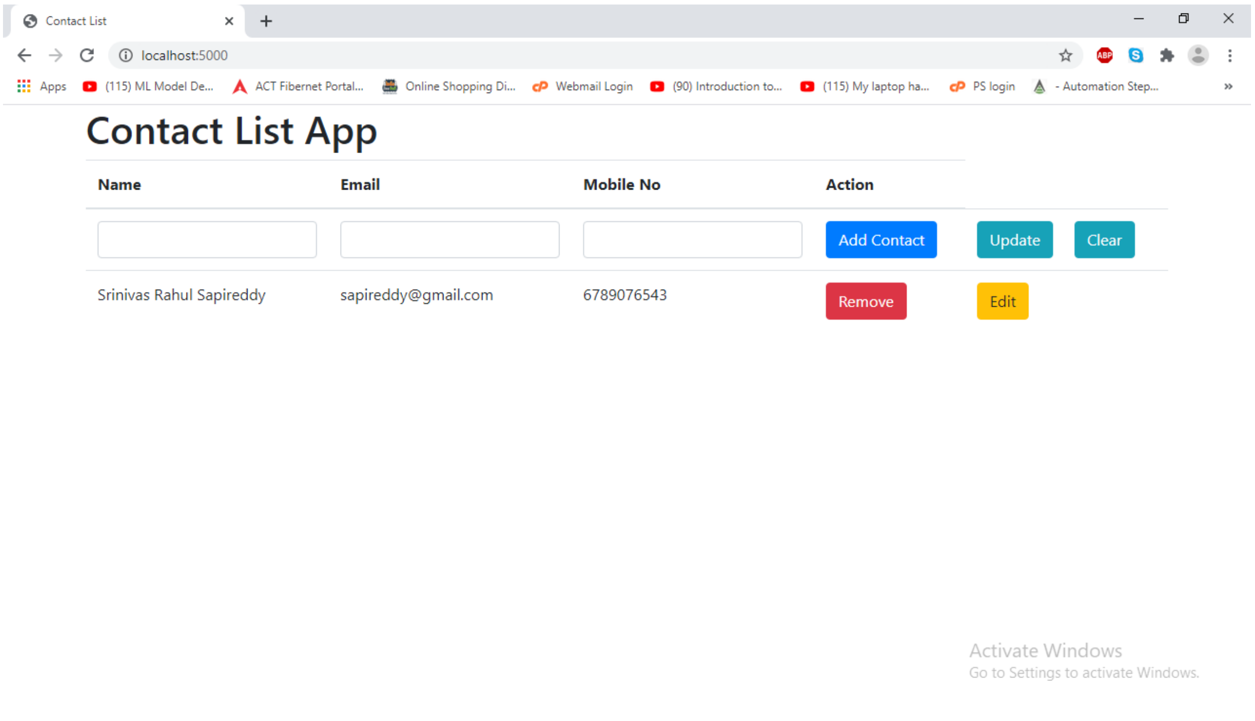
Data Storing in the MongoDB Database shown in the console


Contact List
localhost:5000
Apps (115) ML Model De... ACT Fibernet Portal... Online Shopping Di... Webmail Login (90) Introduction to... (115) My laptop ha... PS login - Automation Step...
Add Contact
Update
Clear
Remove
Edit
Activate Windows
Go to Settings to activate Windows.

Contact List App

Name	Email	Mobile No	Action
Sapireddy	sapireddy@gmail.com	6789076543	Remove Edit

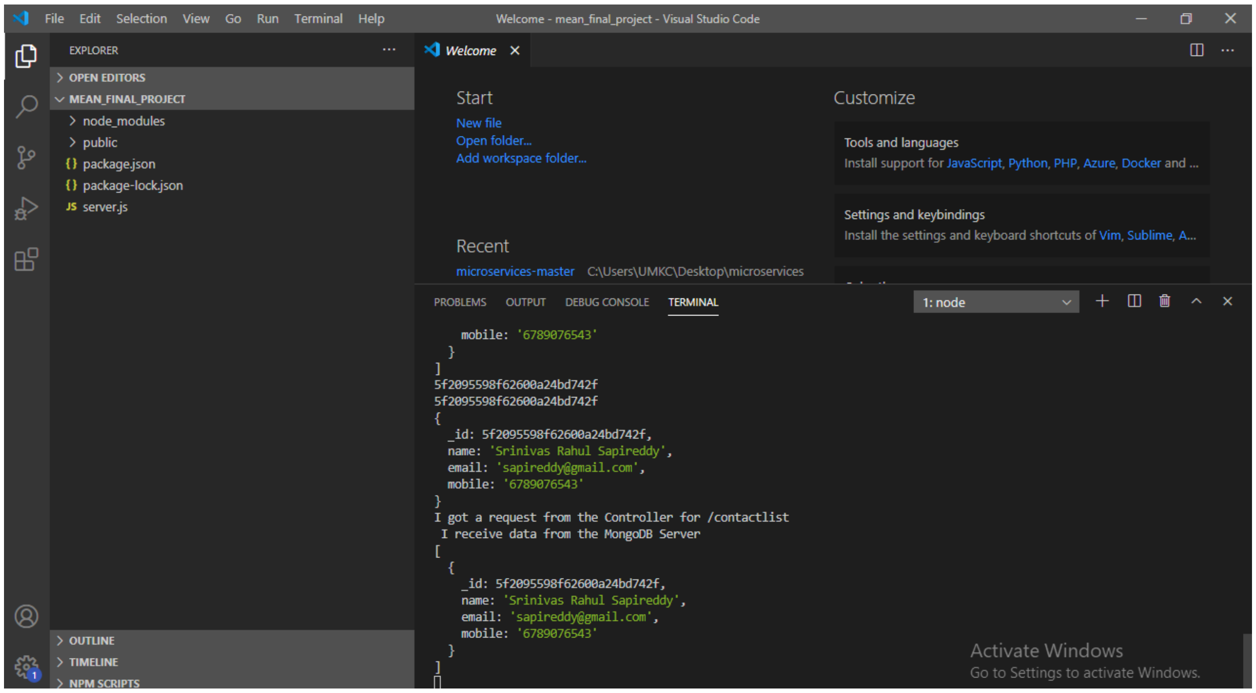
Removed 2 contacts to test delete functionality



Contact List App

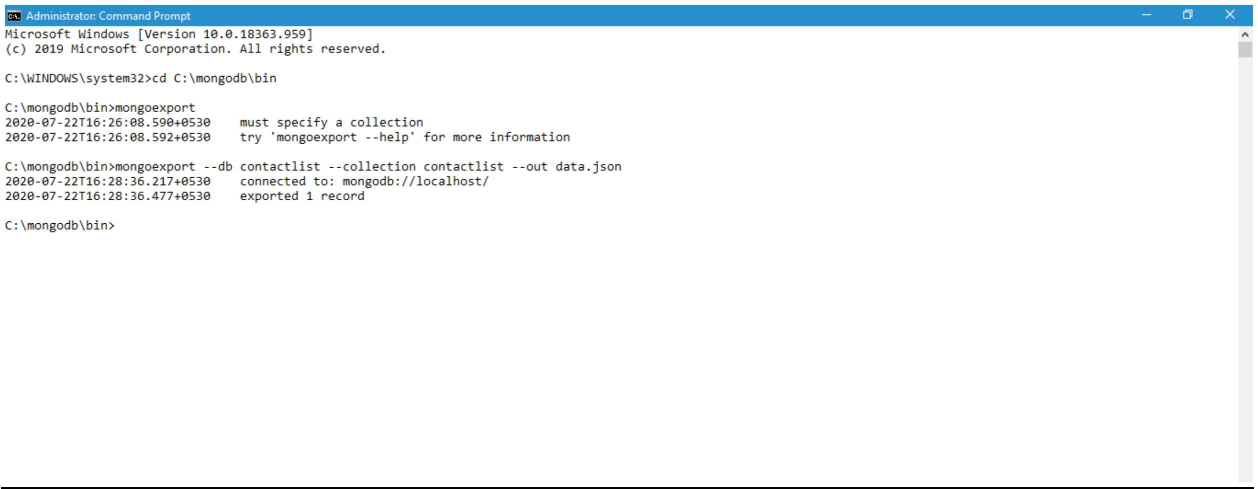
Name	Email	Mobile No	Action		
			Add Contact	Update	Clear
Srinivas Rahul Sapireddy	sapireddy@gmail.com	6789076543	Remove	Edit	

Updated the existing contact



```
    mobile: '6789076543'
  }
]
5f2095598f62600a24bd742f
5f2095598f62600a24bd742f
{
  _id: 5f2095598f62600a24bd742f,
  name: 'Srinivas Rahul Sapireddy',
  email: 'sapireddy@gmail.com',
  mobile: '6789076543'
}
I got a request from the Controller for /contactlist
 I receive data from the MongoDB Server
[
  {
    _id: 5f2095598f62600a24bd742f,
    name: 'Srinivas Rahul Sapireddy',
    email: 'sapireddy@gmail.com',
    mobile: '6789076543'
  }
]
```

Console showing updated data



```
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\mongodb\bin

C:\mongodb\bin>mongoexport
2020-07-22T16:26:08.590+0530    must specify a collection
2020-07-22T16:26:08.592+0530    try 'mongoexport --help' for more information

C:\mongodb\bin>mongoexport --db contactlist --collection contactlist --out data.json
2020-07-22T16:28:36.217+0530    connected to: mongodb://localhost/
2020-07-22T16:28:36.477+0530    exported 1 record

C:\mongodb\bin>
```

Importing Mongo DB Data into JSON

[9] Conclusion

This report is about creating MEAN Stack CURD Application. This application is created using Angular as front end. This project report covers essential topics used for building MEAN Stack Application using Angular. A contact list application created which is used to insert new contacts, delete existing contacts and updating the existing contacts in the application.

Knowledge learnt doing this project/ Course:

(a) Installing required software's for MEAN Stack Development.
(b) How Mongo DB, Angular, Express, NodeJS work together.
(c) Creating an Contact list MEAN Stack Application
(d) Creating RESTFul APIs
(e) Building End to End Web Application using MEAN Stack.

[10] Bibliography and Referencing

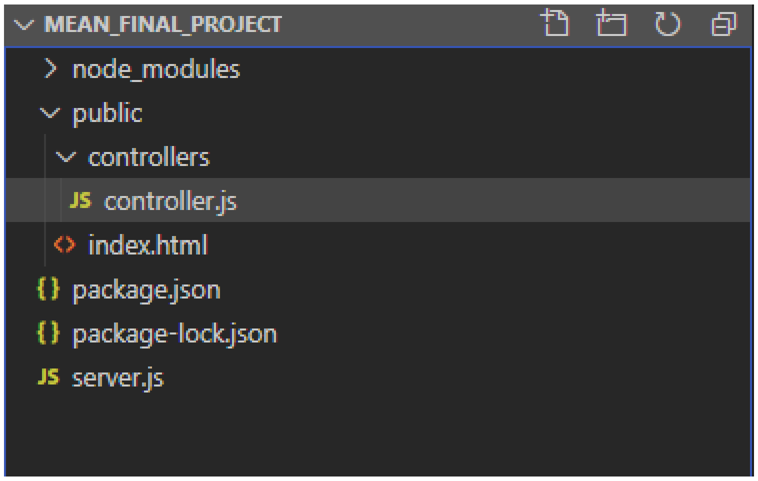
[1] https://dzone.com/articles/mean-stack-amp-startups-are-they-made-for-each-oth

[2] https://www.guru99.com/mean-stack-developer.html

[3] Other Sources: YouTube

CODE

Project Structure


```
MEAN_FINAL_PROJECT
  > node_modules
  v public
    v controllers
      controller.js
    index.html
  package.json
  package-lock.json
  server.js
```

Server.js Code

```
const express = require("express");
const app = express();
const port = 5000;
const path = require("path");

const mongojs = require("mongojs");
const db = mongojs("contactlist", ["contactlist"]);

const bodyParser = require("body-parser");

app.listen(port);

app.use("/", function(req,res,next){
    next();
})

app.use(express.static(path.join(__dirname,"public")));

app.use(bodyParser.json()); // middleware

app.get("/", function(req,res){
    res.sendFile(__dirname + "/index.html");

})
```

```
app.get('/contactlist', function(req,res){
    console.log("I got a request from the Controller for /contactlist");

    db.contactlist.find(function(err,docs){
        console.log(" I receive data from the MongoDB Server");
        console.log(docs);
        res.json(docs);
    })



})


app.post("/contactlist", function(req,res){
    console.log("Yes I received the New Data ...");
    console.log(req.body);

    db.contactlist.insert(req.body, function(err, doc){
        res.json(doc);
        })
})

app.delete("/contactlist/:id", function(req,res){
    var id = req.params.id;
    console.log(id);

db.contactlist.remove({_id: mongojs.ObjectID(id)}, function(err,docs){
    res.json(docs);
})

})

app.get("/contactlist/:id", function(req,res){
    var id = req.params.id;
    console.log(id);

    db.contactlist.findOne({_id: mongojs.ObjectID(id)}, function(err,doc){
        res.json(doc);
        })
})

app.put("/contactlist/:id", function(req,res){
    var id = req.params.id;
    console.log(id);
    db.contactlist.findAndModify({query: {_id: mongojs.ObjectID(id)},
    update: {$set : {name: req.body.name, email: req.body.email, mobile: req.body
.mobile}},
    new: true}, function(err,docs){
        console.log(docs);

        res.json(docs);
    })
})


console.log("Server is Running");
```

controller.js code

```
angular.module('myApp', []).controller('appCtrl', function($scope,$http)
{

var refresh = function(){

    $http.get('/contactlist').success(function(response){

        console.log('Yes I received the data back');



        $scope.contactlist = response;
        $scope.contact = "";

    });
}

    refresh();

    $scope.addContact = function(){
        console.log($scope.contact);

        $http.post("/contactlist", $scope.contact).success(function(response){
            console.log(response);
            refresh();
        });
    }


    $scope.remove = function(id){
        console.log(id);

        $http.delete("/contactlist/" + id).success(function(response){
            refresh();
        });
    }

    $scope.edit = function(id) {
        console.log(id);

        $http.get("/contactlist/" + id).success(function(response){
            $scope.contact = response;
        });
    }

    $scope.update = function() {
        var id = $scope.contact._id;
        console.log(id);

        $http.put("/contactlist/" + id, $scope.contact).success(function(response
){
            refresh();
        });
    }

    $scope.deselect = function(){
        $scope.contact = "";
    }

})
```

index.html code

```
<!DOCTYPE html>
<html ng-app="myApp" ng-controller="appCtrl">
<head>
    <title>Contact List</title>
</head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css" integrity="sha384-
```

```
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="an
onymous">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js
"></script>

<body>

    <div class="container">

    <h1>Contact List App</h1>

    <table class="table">
        <thead>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Mobile No</th>
                <th>Action</th>
            </tr>
        </thead>

    <tbody>
     <tr>
        <td><input class="form-control" ng-model="contact.name"</td>
        <td><input class="form-control" ng-model="contact.email"</td>
        <td><input class="form-control" ng-model="contact.mobile"</td>
        <td><button class="btn btn-primary" ng-
click="addContact()">Add Contact</button></td>

        <td><button class="btn btn-info" ng-
click="update()">Update</button> &nbsp &nbsp
        <button class="btn btn-info" ng-click="deselect()">Clear</button></td>
    </tr>

        <tr ng-repeat = "contact in contactlist">
            <td>{{contact.name}}</td>
            <td>{{contact.email}}</td>
            <td>{{contact.mobile}}</td>

            <td> <button class="btn btn-danger" ng-
click = "remove(contact._id)">Remove</button></td>
            <td> <button class="btn btn-warning" ng-
click = "edit(contact._id)">Edit</button></td>
        </tr>
    </tbody>
    </table>



    </div>

    <script src="controllers/controller.js"></script>

</body>
</html>
```