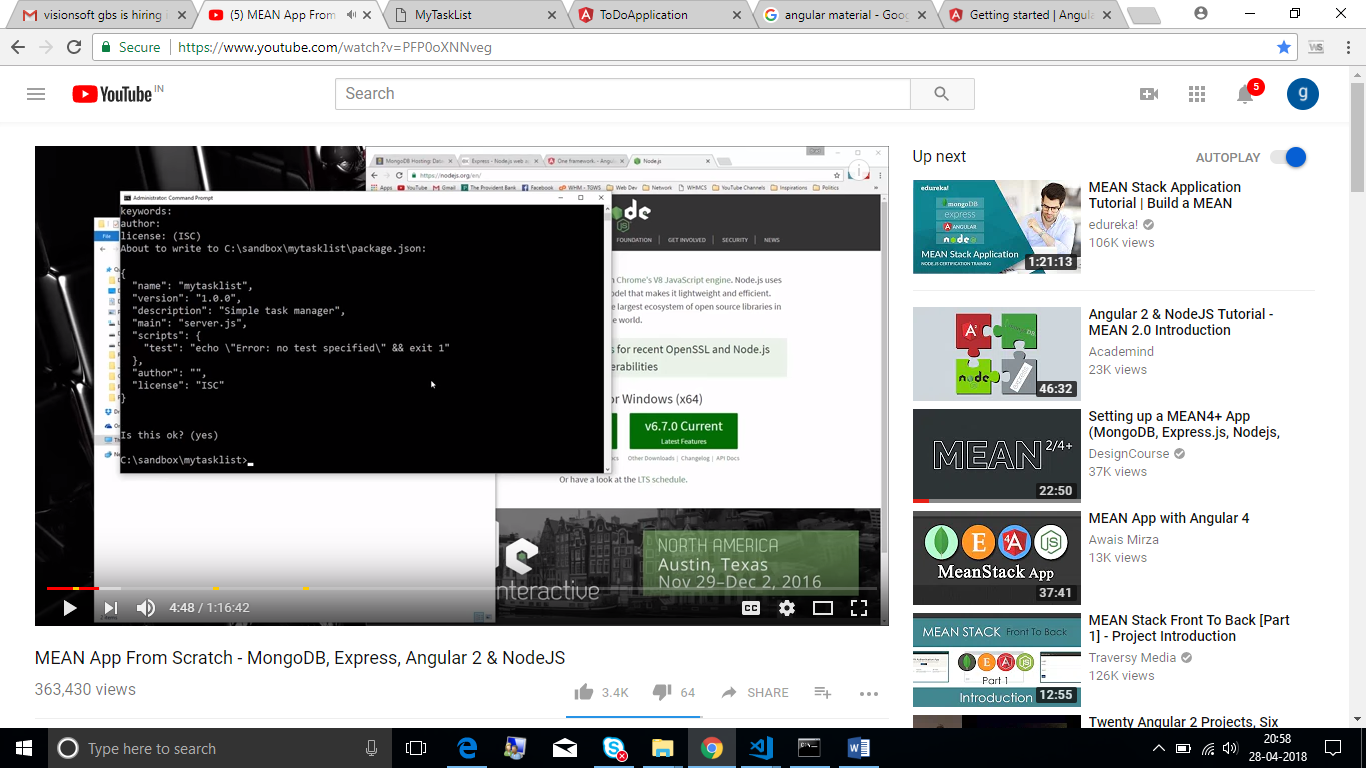
**TODO APPLICATION WITH EXPRESS , MONGO DB, ANGULAR 2 AND NODE JS**

* MongoDB : using mlab(remote implementation of MongoDB; so no need to install MongoDB on local machine).
* Express: install the express in the application.It is a framework used to develop web application that supports routing, http request.
* Angular: It is used for frontend(client side UI).
* MongoJs: Use mongojs which will interact with the mongodb database which do insert, find operations.
* NodeJs
* Create a project folder: named as **MyTodoApplication**
* At the command prompt: run the commad as **npm init** within the **MyTodoApplication**

to generate the **package.json** within the project folder.

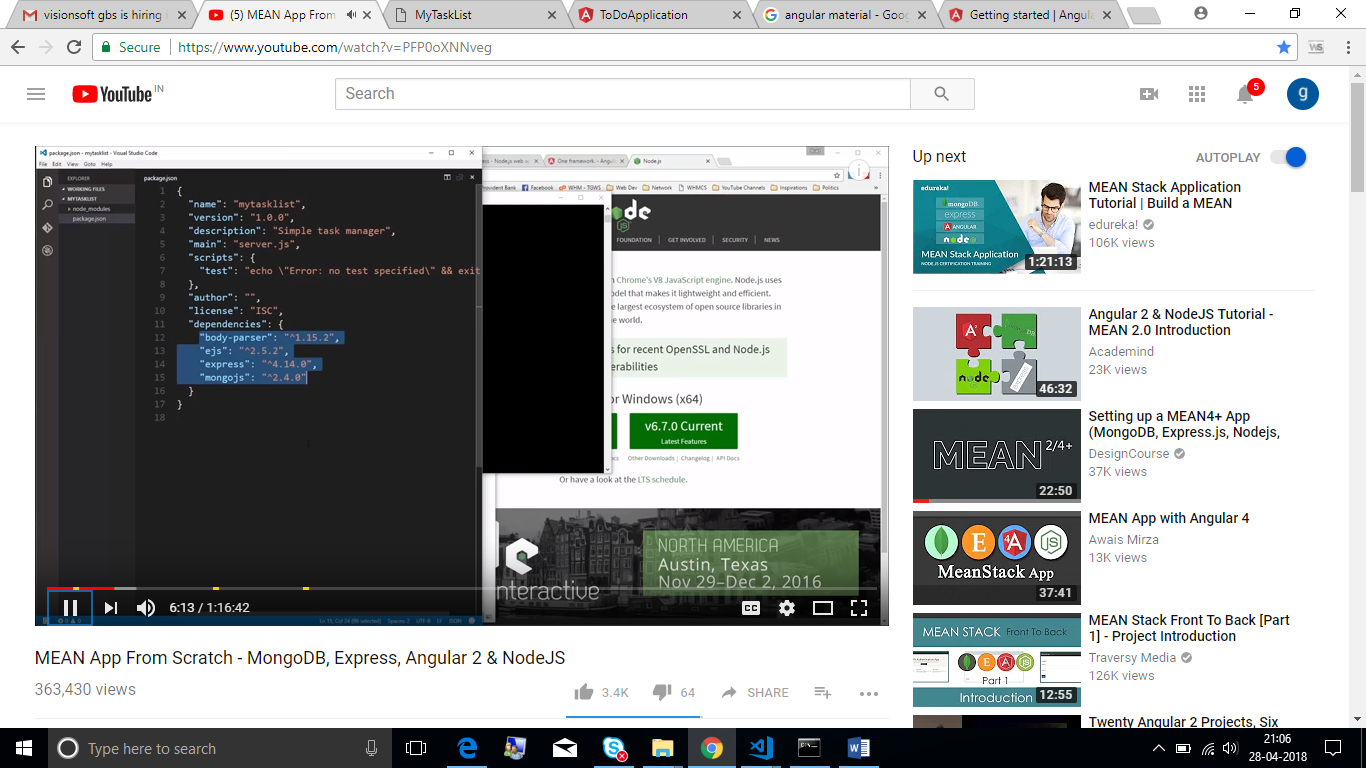


* Open the editor of your choice(here using Visual Studio code) and open the project folder within the editor.
* Now install all the dependencies.

In the command line; move into the project folder and run the following command:

npm install express body-parser ejs mongojs --save

which will generate project dependencies in the package.json file of the project folder as shown in the screenshot.



* Now create the **server.js** file.

server.js is the main backend server file.

The content of the server.js file will be as follows:

var express = require('express');

var path = require('path');

var bodyParser = require('body-parser');

var index = require('./routes/index');

var tasks = require('./routes/tasks');

var port = 4000;

var app = express();

//View Engine

app.set('views', path.join(\_\_dirname, 'views'));

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

app.engine('html', require('ejs').renderFile);

// Set Static Folder

app.use(express.static(path.join(\_\_dirname, 'client')));

// Body Parser MW

app.use(bodyParser.json());

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

app.use('/', index);

app.use('/api', tasks);

app.listen(port, function(){

console.log('Server started on port '+port);

});

* Now create a folder **routes** and create two files **index.js** and **tasks.js** within the folder **routes**.

Index.js :

var express = require('express');

var router = express.Router();

router.get('/', function(req, res, next){

res.render('index.html');

});

module.exports = router;

tasks.js:

var express = require('express');

var router = express.Router();

var mongojs = require('mongojs');

var db = mongojs('mongodb://admin:admin@ds159509.mlab.com:59509/mytasklist', ['tasks']);

// Get All Tasks

router.get('/tasks', function(req, res, next){

db.tasks.find(function(err, tasks){

if(err){

res.send(err);

}

res.json(tasks);

});

});

// Get Single Task

router.get('/task/:id', function(req, res, next){

db.tasks.findOne({\_id: mongojs.ObjectId(req.params.id)}, function(err, task){

if(err){

res.send(err);

}

res.json(task);

});

});

//Save Task

router.post('/task', function(req, res, next){

var task = req.body;

if(!task.title || !(task.isDone + '')){

res.status(400);

res.json({

"error": "Bad Data"

});

} else {

db.tasks.save(task, function(err, task){

if(err){

res.send(err);

}

res.json(task);

});

}

});

// Delete Task

router.delete('/task/:id', function(req, res, next){

db.tasks.remove({\_id: mongojs.ObjectId(req.params.id)}, function(err, task){

if(err){

res.send(err);

}

res.json(task);

});

});

// Update Task

router.put('/task/:id', function(req, res, next){

var task = req.body;

var updTask = {};

if(task.isDone){

updTask.isDone = task.isDone;

}

if(task.title){

updTask.title = task.title;

}

if(!updTask){

res.status(400);

res.json({

"error":"Bad Data"

});

} else {

db.tasks.update({\_id: mongojs.ObjectId(req.params.id)},updTask, {}, function(err, task){

if(err){

res.send(err);

}

res.json(task);

});

}

});

module.exports = router;

* Now create a **views** and within the views folder create **index.html**.

Index.html

<html>

<head>

<title>MyTaskList</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="bower\_components/bootstrap/dist/css/bootstrap.css">

<link rel="stylesheet" href="styles.css">

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

<!-- 1. Load libraries -->

<!-- Polyfill(s) for older browsers -->

<script src="node\_modules/core-js/client/shim.min.js"></script>

<script src="node\_modules/zone.js/dist/zone.js"></script>

<script src="node\_modules/reflect-metadata/Reflect.js"></script>

<script src="node\_modules/systemjs/dist/system.src.js"></script>

<!-- 2. Configure SystemJS -->

<script src="systemjs.config.js"></script>

<script>

System.import('app').catch(function(err){ console.error(err); });

</script>

</head>

<!-- 3. Display the application -->

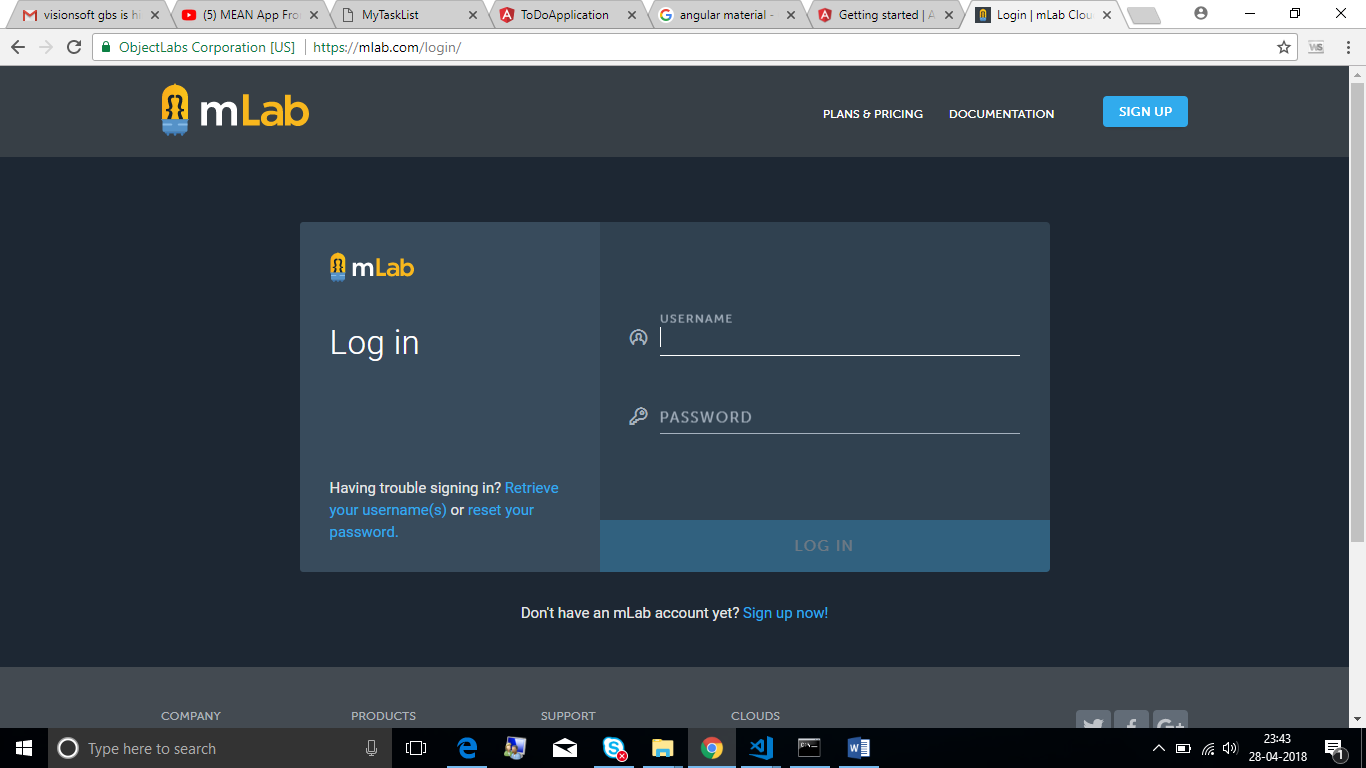
<body>

<my-app>Loading...</my-app>

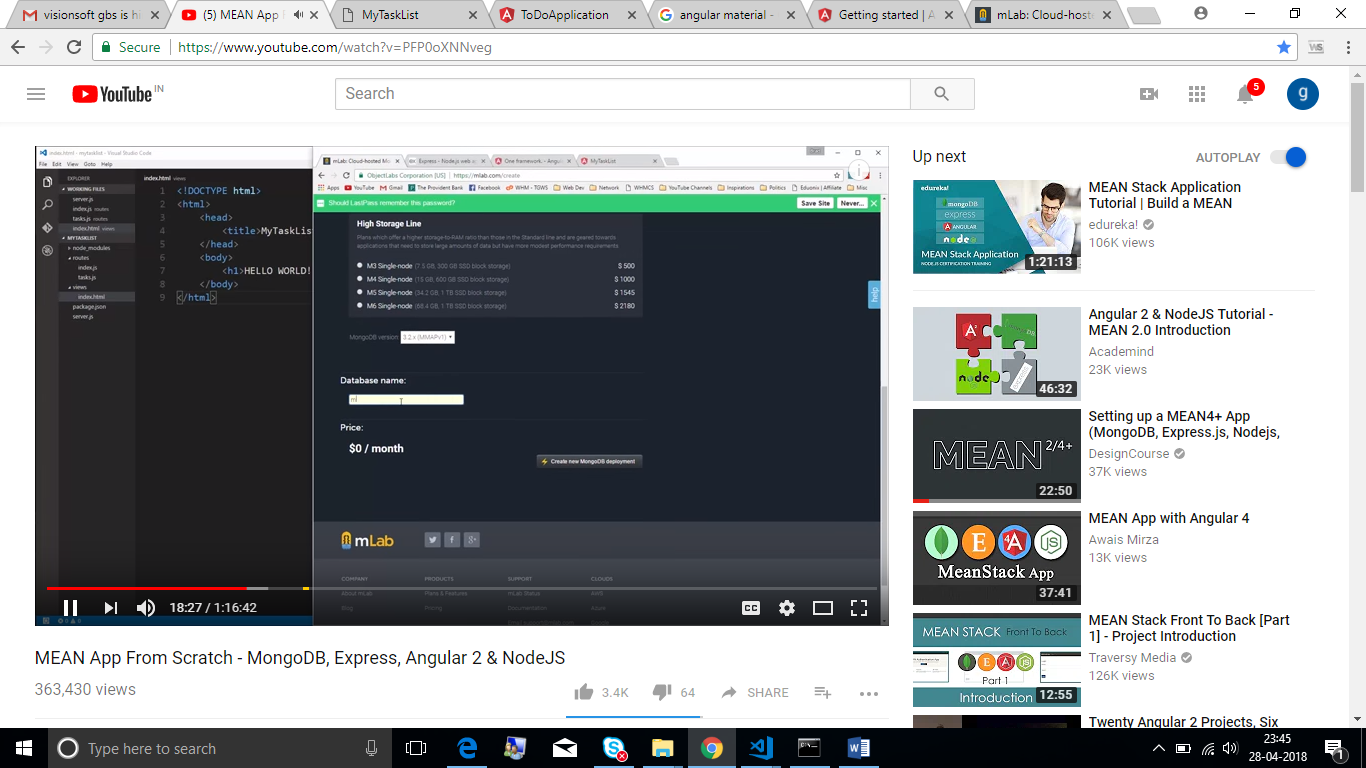
</body>

</html>

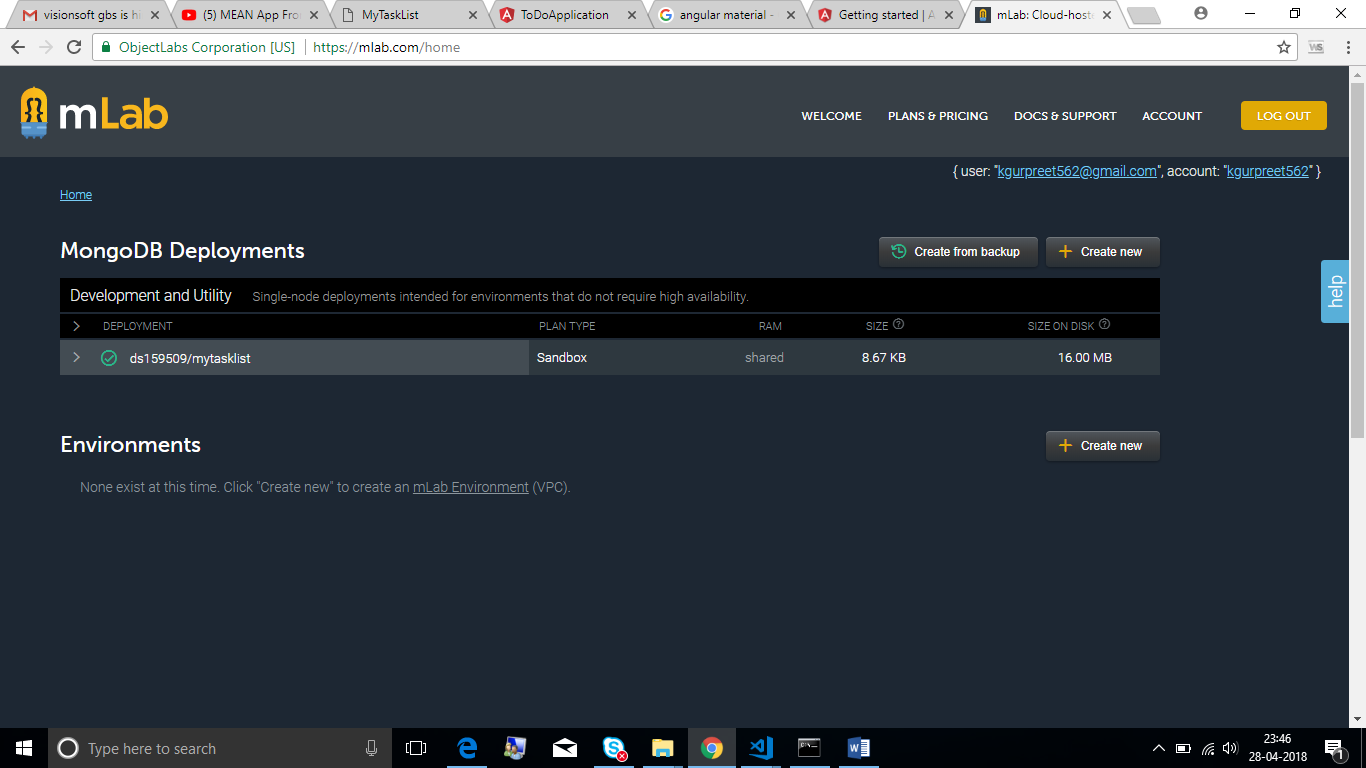
* Now create a account in mlab for accessing mongodb.



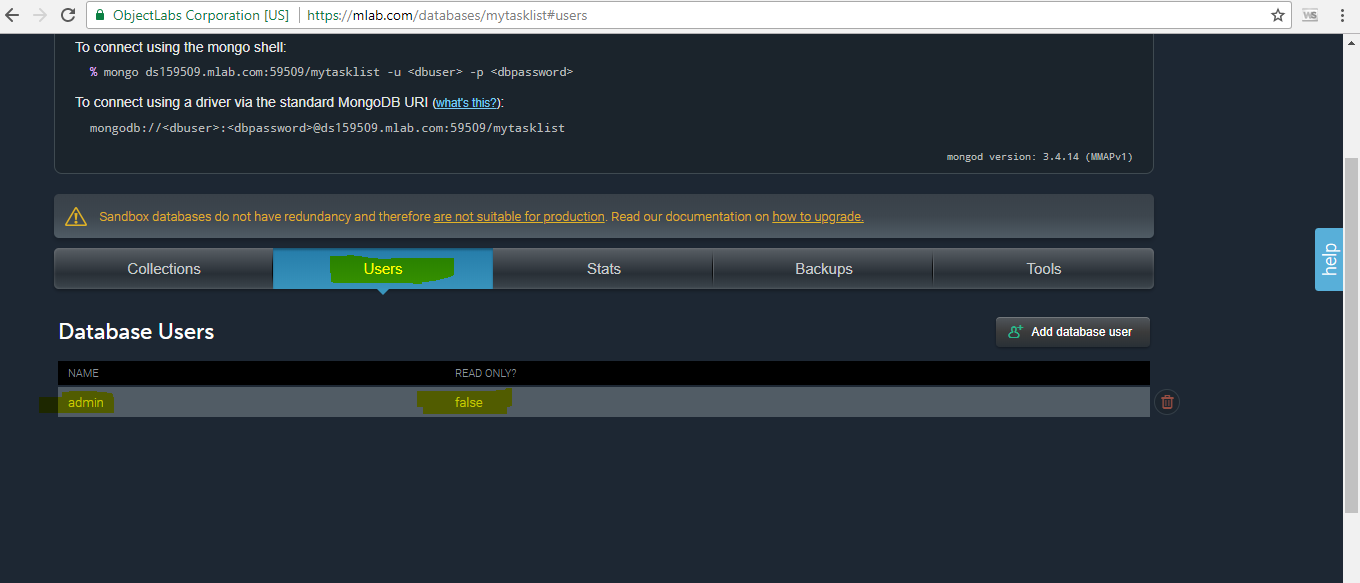
* Now create a database with particular name.



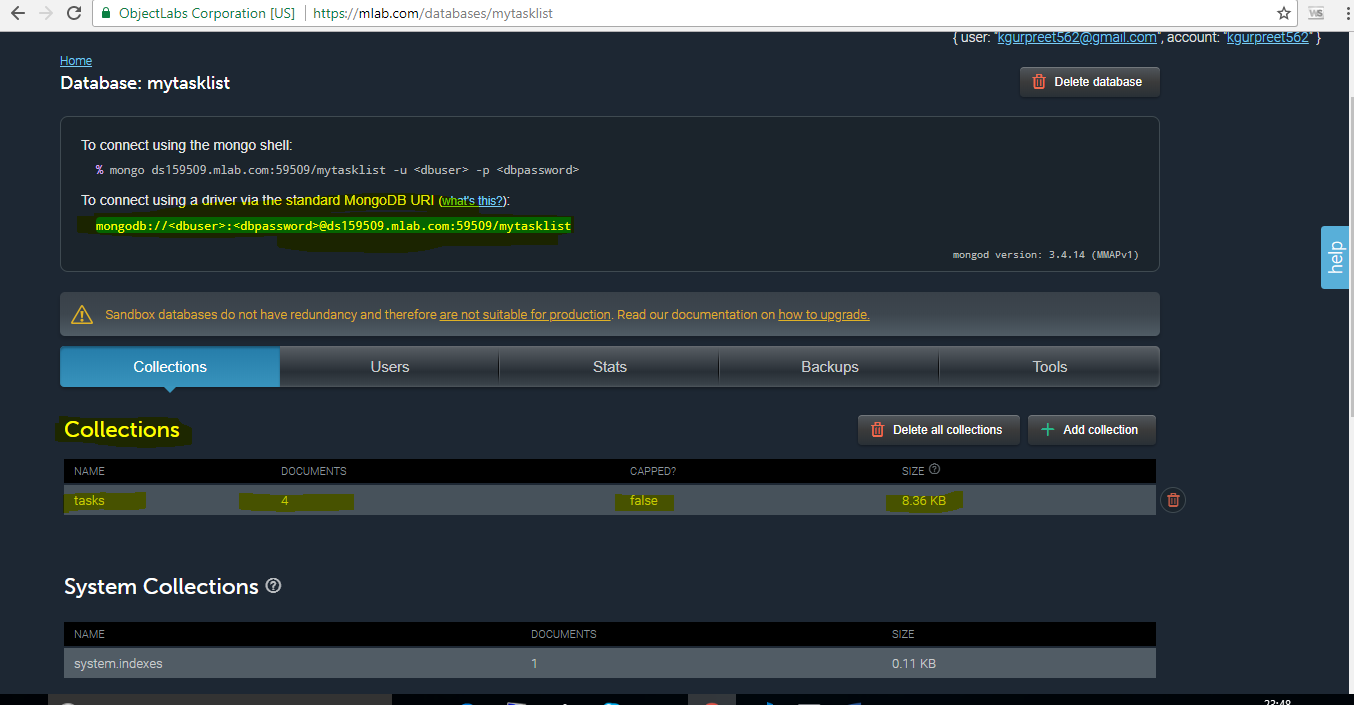
* A database will be created.



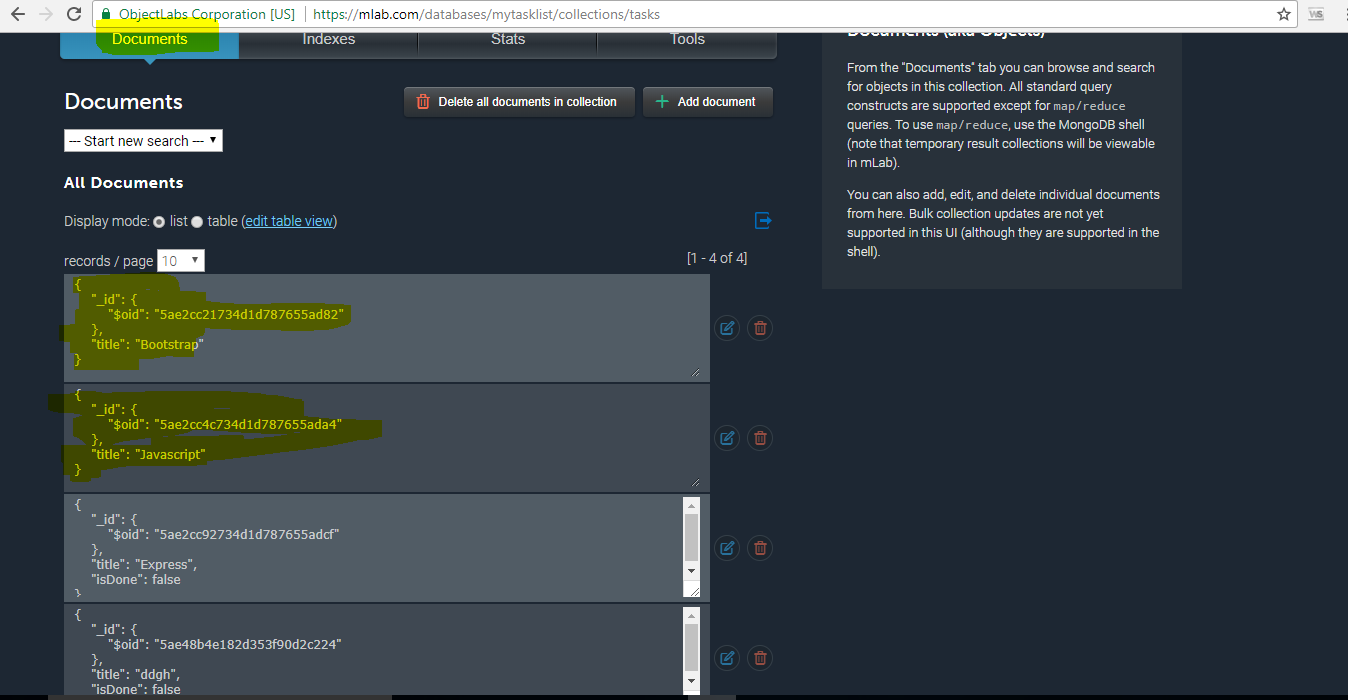
* Create a database user.



* Now create a collection with the documents in the collections as highlighted in the screenshot.



* Create documents within the collection as shown in the screenshot.



* Now create a client folder.

Now create four main config files.

* package.json
* typings.json
* systemjs.config.js
* tsconfig.json

package.json:

{

"name": "mytasklist",

"version": "1.0.0",

"scripts": {

"start": "tsc && concurrently \"tsc -w\" \"lite-server\" ",

"lite": "lite-server",

"postinstall": "typings install",

"tsc": "tsc",

"tsc:w": "tsc -w",

"typings": "typings"

},

"license": "ISC",

"dependencies": {

"@angular/common": "~2.0.1",

"@angular/compiler": "~2.0.1",

"@angular/core": "~2.0.1",

"@angular/forms": "~2.0.1",

"@angular/http": "~2.0.1",

"@angular/platform-browser": "~2.0.1",

"@angular/platform-browser-dynamic": "~2.0.1",

"@angular/router": "~3.0.1",

"@angular/upgrade": "~2.0.1",

"angular-in-memory-web-api": "~0.1.1",

"bootstrap": "^3.3.7",

"core-js": "^2.4.1",

"reflect-metadata": "^0.1.8",

"rxjs": "5.0.0-beta.12",

"systemjs": "0.19.39",

"zone.js": "^0.6.25"

},

"devDependencies": {

"concurrently": "^3.0.0",

"lite-server": "^2.2.2",

"typescript": "^2.0.3",

"typings":"^1.4.0"

}

}

typings.json:

{

"compilerOptions": {

"target": "es5",

"module": "commonjs",

"moduleResolution": "node",

"sourceMap": true,

"emitDecoratorMetadata": true,

"experimentalDecorators": true,

"removeComments": false,

"noImplicitAny": false

}

}

systemjs.config.js:

/\*\*

\* System configuration for Angular samples

\* Adjust as necessary for your application needs.

\*/

(function (global) {

System.config({

paths: {

// paths serve as alias

'npm:': 'node\_modules/'

},

// map tells the System loader where to look for things

map: {

// our app is within the app folder

app: 'app',

// angular bundles

'@angular/core': 'npm:@angular/core/bundles/core.umd.js',

'@angular/common': 'npm:@angular/common/bundles/common.umd.js',

'@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',

'@angular/platform-browser': 'npm:@angular/platform-browser/bundles/platform-browser.umd.js',

'@angular/platform-browser-dynamic': 'npm:@angular/platform-browser-dynamic/bundles/platform-browser-dynamic.umd.js',

'@angular/http': 'npm:@angular/http/bundles/http.umd.js',

'@angular/router': 'npm:@angular/router/bundles/router.umd.js',

'@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',

// other libraries

'rxjs': 'npm:rxjs',

'angular-in-memory-web-api': 'npm:angular-in-memory-web-api',

},

// packages tells the System loader how to load when no filename and/or no extension

packages: {

app: {

main: './main.js',

defaultExtension: 'js'

},

rxjs: {

defaultExtension: 'js'

},

'angular-in-memory-web-api': {

main: './index.js',

defaultExtension: 'js'

}

}

});

})(this);

tsconfig.json:

{

"compilerOptions": {

"target": "es5",

"module": "commonjs",

"moduleResolution": "node",

"sourceMap": true,

"emitDecoratorMetadata": true,

"experimentalDecorators": true,

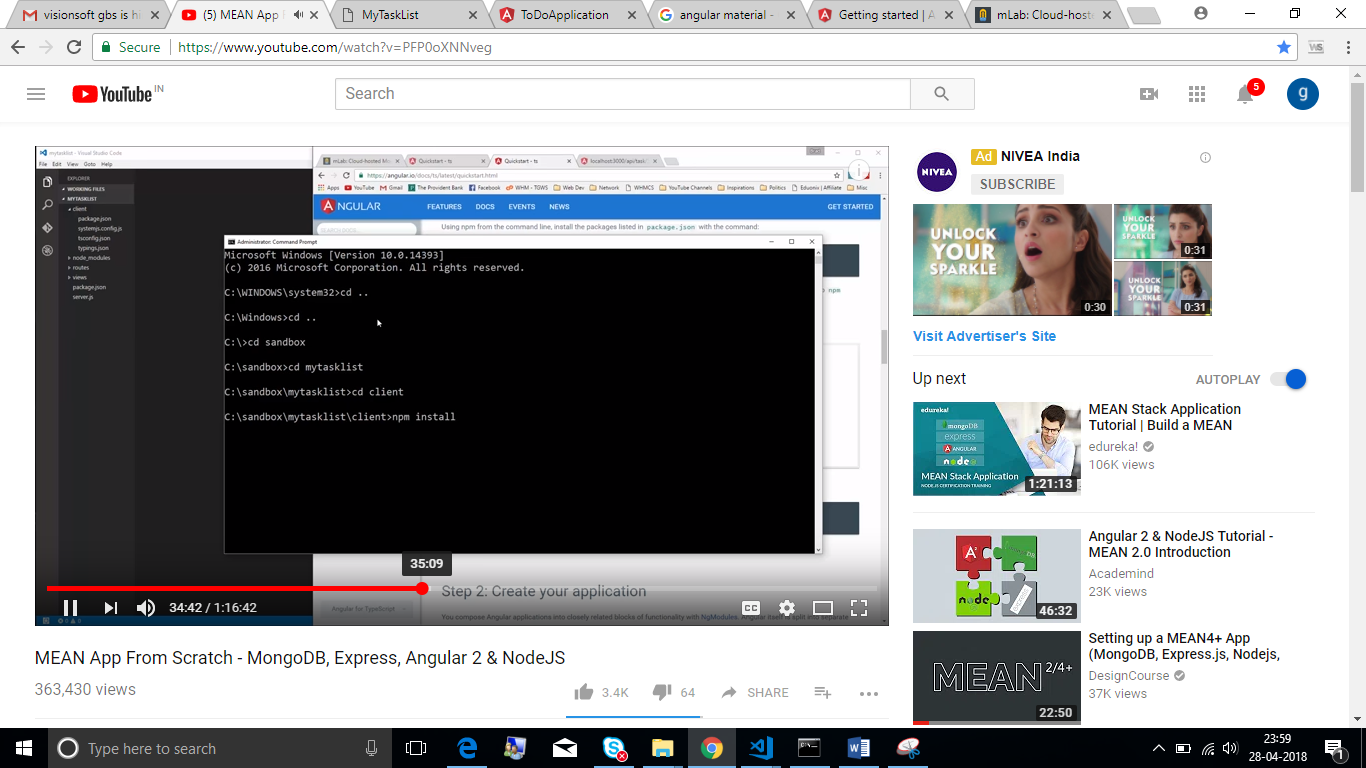
"removeComments": false,

"noImplicitAny": false

}

}

* Now navigate to the **client** folder and run **npm install**.



* Now inside the client create a app folder.

Inside the app folder; create the following files.

App.module.ts:

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import {HttpModule} from '@angular/http';

import {FormsModule} from '@angular/forms';

import {AppComponent} from './app.component';

import {TasksComponent} from './components/tasks/tasks.component';

@NgModule({

imports: [ BrowserModule, HttpModule, FormsModule],

declarations: [AppComponent, TasksComponent],

bootstrap: [AppComponent]

})

export class AppModule { }

app.component.ts:

import { Component } from '@angular/core';

import {TaskService} from './services/task.service';

@Component({

moduleId: module.id,

selector: 'my-app',

templateUrl: 'app.component.html',

providers:[TaskService]

})

export class AppComponent { }

app.component.html

<div>

<div class="container">

<h1> MyTaskList </h1>

</div>

<tasks></tasks>

</div>

Main.ts:

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app.module';

import 'hammerjs';

const platform = platformBrowserDynamic();

platform.bootstrapModule(AppModule);

* Now create a folder component within the app folder.

Create folder component-> create folder tasks->create files

* tasks.component.ts
* tasks.component.html

tasks.component.ts:

import { Component } from '@angular/core';

import {TaskService} from '../../services/task.service';

import {Task} from '../../../Task';

@Component({

moduleId: module.id,

selector: 'tasks',

templateUrl: 'tasks.component.html'

})

export class TasksComponent {

tasks: Task[];

title: string;

constructor(private taskService:TaskService){

this.taskService.getTasks()

.subscribe(tasks => {

this.tasks = tasks;

});

}

addTask(event){

event.preventDefault();

var newTask = {

title: this.title,

isDone: false

}

this.taskService.addTask(newTask)

.subscribe(task => {

this.tasks.push(task);

this.title = '';

});

}

deleteTask(id){

var tasks = this.tasks;

this.taskService.deleteTask(id).subscribe(data => {

if(data.n == 1){

for(var i = 0;i < tasks.length;i++){

if(tasks[i].\_id == id){

tasks.splice(i, 1);

}

}

}

});

}

updateStatus(task){

var \_task = {

\_id:task.\_id,

title: task.title,

isDone: !task.isDone

};

this.taskService.updateStatus(\_task).subscribe(data => {

task.isDone = !task.isDone;

});

}

}

tasks.component.html:

<form class="well" (submit)="addTask($event)">

<div class="form-group">

<input type="text" [(ngModel)]="title" name="title" class="form-control" placeholder="Add Task...">

</div>

</form>

<div class="task-list">

<div \*ngFor="let task of tasks">

<div class="col-md-1">

<input type="checkbox" [checked]="task.isDone" (click)="updateStatus(task)">

</div>

<div class="col-md-7">

{{task.title}}

</div>

<div class="col-md-4">

<input type="button" (click)="deleteTask(task.\_id)" value="Delete" class="btn btn-danger">

</div>

<br><br>

</div>

</div>

* now create a folder services within the app folder and create file task.service.ts

import {Injectable} from '@angular/core';

import {Http, Headers} from '@angular/http';

import 'rxjs/add/operator/map';

@Injectable()

export class TaskService{

constructor(private http:Http){

console.log('Task Service Initialized...');

}

getTasks(){

return this.http.get('/api/tasks')

.map(res => res.json());

}

addTask(newTask){

var headers = new Headers();

headers.append('Content-Type', 'application/json');

return this.http.post('/api/task', JSON.stringify(newTask), {headers: headers})

.map(res => res.json());

}

deleteTask(id){

return this.http.delete('/api/task/'+id)

.map(res => res.json());

}

updateStatus(task){

var headers = new Headers();

headers.append('Content-Type', 'application/json');

return this.http.put('/api/task/'+task.\_id, JSON.stringify(task), {headers: headers})

.map(res => res.json());

}

}

**FINAL OUTPUT:**

