Contents

[Introduction 3](#_Toc503269636)

[API 4](#_Toc503269637)

[Create Model 4](#_Toc503269638)

[Create Service 7](#_Toc503269639)

[Create Controller 8](#_Toc503269640)

[Create Route 9](#_Toc503269641)

[Angular 2 Core, Shared and Routing Components 10](#_Toc503269642)

[Create Model 10](#_Toc503269643)

[Create Service 12](#_Toc503269644)

[.spec 14](#_Toc503269645)

[Create Resolve 14](#_Toc503269646)

[Module Imports 15](#_Toc503269647)

[Routing 16](#_Toc503269648)

[Angular 2 Component 16](#_Toc503269649)

[Module Imports 16](#_Toc503269650)

# Introduction

This document provides walkthrough of the stages required to implement a MEAN stack project. It will demonstrate an application enabling basic CRUD functionality as well as sporting file upload and record search capability.

Please note that the code examples used in this documentation are highlighted to indicate the current segment of code. Code that is greyed out can be ignored and is there for the purposes of setting context and completeness.

# Prerequisites

This guide assumes that the project adheres to the file structure as set out in the following URL reference.

It also assumes that the following docker container is present:-

docker run -p 27017:27017 -d -v ~/Documents/mongo-data:/data/db

--name mongo-db mongo

To install and build the project, execute the following steps on the project root at the Command Prompt:-

npm install

ng build

To start the project in the browser, execute the following step from a terminal window:-

node server.js

Open up a browser window and enter the following URL to display the application:-

**http://localhost:3000/**

# CRUD API

## Install

1. Install the **mongodb** npm module:-

npm install mongodb –-save

1. Install the **mongoose** npm module:-

npm install mongoose –-save

## Create Model

1. Create **record.js** file in the following location:-

**<proj\_root>/server/src/models/record.js**

1. Open the file and enter the following code:-

var mongoose = require('mongoose');

// mongoose schema

var recordSchema = mongoose.Schema({

firstName: {

type: String,

required: true

},

lastName: {

type: String,

required: true

},

emailAddress: {

type: String,

required: false

},

});

// compile schema into model (class)

var record = mongoose.model('Record', recordSchema);

module.exports = record;

## Create Service

1. Create **portalService.js** in the following location:-

**<proj\_root>/server/src/portal/portalService.js**

1. Open the file and enter the following code:-

const mongoose = require('mongoose');

const Record = require('../../models/record.js');

var exports = module.exports = {};

1. Enter the following code for the **findAllRecords()** function:-

exports.findAllRecords = () => Record.find({})

.exec();

1. Enter the following code for the **createRecord()** function:-

exports.createRecord = function (record) {

// compile schema into model (class)

var collection = mongoose.model('Record');

// create document

var doc = new collection(record);

// save document to MongoDB

return doc.save(function (err, record) {

if (err) return console.error(err);

console.log('Record Created!');

});

};

1. Enter the following code for the **updateRecord()** function:-

exports.updateRecord = function (record, onErr, onSuccess) {

var collection = mongoose.model('Record');

var query = { \_id: record.\_id };

collection.findOneAndUpdate(query, { firstName: record.firstName, lastName: record.lastName, emailAddress: record.emailAddress }, function (err, data) {

if (err) return onErr;

return onSuccess;

});

};

1. Enter the following code for the **deleteRecord()** function:-

exports.deleteRecord = function (id, onErr, onSuccess) {

var collection = mongoose.model('Record');

collection.findOneAndRemove({ '\_id': id }, function (err, data) {

if (err) return onErr;

return onSuccess;

});

};

## Create Controller

1. Create **portalController.js** in the following location:-

**<proj\_root>/server/src/portal/ portalController.js**

1. Open the file and enter the following code:-

const portalService = require('./portalService');

var exports = module.exports = {};

1. Enter the following code for the **findAllRecords()** function:-

exports.findAllRecords = function (req, res) {

portalService.findAllRecords().then(data => res.status(200).send(data));

}

1. Enter the following code for the **createRecord()** function:-

exports.createRecord = function (req, res) {

console.log('portal controller: create record');

var record = req.body ? req.body : undefined;

if (!record) {

res.status(400).send("Empty record");

} else {

portalService.createRecord(record)

.then(data => res.status(200).send(data))

.catch('an error occurred');

}

}

1. Enter the following code for the **updateRecord()** function:-

exports.updateRecord = function (req, res) {

console.log('portal controller: update record');

var record = req.body ? req.body : undefined;

if (!record) {

res.status(400).send("Empty record");

} else {

var onErr = (err) => {

res.status(500).send(err);

};

var onSuccess = (data) => {

var response = res.status(200);

response.setHeader('Expires', '-1');

response.send(data);

console.log('onSuccess: Record Saved!');

};

portalService.updateRecord(record, onErr, onSuccess);

};

};

1. Enter the following code for the **deleteRecord()** function:-

exports.deleteRecord = function (req, res) {

console.log('portal controller: delete record');

var id = req.params ? req.params.id : undefined;

if (!id) {

res.status(400).send("Delete Operation Error: Invalid Identifier");

} else {

var onErr = (err) => {

res.status(500).send(err);

};

var onSuccess = (data) => {

var response = res.status(200);

response.redirect('/');

console.log('onSuccess: Record Deleted!');

};

portalService.deleteRecord(id, onErr, onSuccess);

}

}

## Create Route

1. Create **portalRoutes.js** in the following location:-

**<proj\_root>/server/src/portal/portalRoutes.js**

1. Open the file and enter the following code:-

const express = require('express');

const router = express.Router();

const portalController = require('./portalController');

1. Add the following routes:-

router.get('/', portalController.findAllRecords);

router.post('/', portalController.createRecord);

router.patch('/', portalController.updateRecord);

router.delete('/:id', portalController.deleteRecord);

module.exports = router;

## Add Route

1. Add the following to **server.js**:-

// Get API routes

const portalRoutes = require('./server/src/portal/portalRoutes.js');

// Set API routes

app.use('/api/portal', portalRoutes);

# Angular 2 Core, Shared and Routing Components

## Create Model

1. Create a **record.model.ts** file in the following location:-

**<proj\_root>/webapp/app/core/portal/record.model.ts**

1. Open the file and enter the following code:-

export class Record {

public \_id: string;

public firstName: string;

public lastName: string;

public emailAddress: string;

constructor(input: Object) {

this.\_id = input["\_id"];

this.firstName = input["firstName"];

this.lastName = input["lastName"];

this.emailAddress = input["emailAddress"];

}

}

## Create Service

1. Create a **record.service.ts** file in the following location:-

**<proj\_root>/webapp/app/shared/service/record.service.ts**

1. Open the file and enter the following code:-

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

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

import { HttpClient, HttpParams } from '@angular/common/http'

import { Record } from '../../core/record.model';

import { Observable } from 'rxjs/Rx';

@Injectable()

export class RecordService {

public record: any;

constructor(

private http: Http,

private httpClient: HttpClient

) { }

1. Enter the following code for the **getRecords()** function:-

getRecords(): Observable<any> {

let headers = new Headers();

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

let options = new RequestOptions({ headers: headers });

return this.http.get('/api/portal', options)

.map(response => (response.json() as Object[]).map(record => new Record(record)))

.catch(error => Observable.throw(error.json().error || 'Server error'))

}

}

1. Enter the following code for the **createRecords()** function:-

createRecord(record: Record): Observable<Record> {

let headers = new Headers();

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

let options = new RequestOptions({ headers: headers });

return this.http.post('/api/portal', record, options).map(res => new Record(res.json()))

}

1. Enter the following code for the **updateRecord()** function:-

updateRecord(record: Record): Observable<Record> {

// set json header information

let headers = new Headers();

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

// gather request options

let options = new RequestOptions({ headers: headers });

// fetch form data and submit to MongoDB api service

return this.http.patch('/api/portal', record, options).map(res => new Record(res.json()))

}

1. Enter the following code for the **deleteRecord()** function:-

deleteRecord(id: string): Observable<Record> {

return this.http.delete('api/portal/' + id).map(res => res.json());

}

## Create Component

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Open the file and enter the following code imports:-

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

import { ActivatedRoute, Params } from '@angular/router';

import { Record } from '../core/record.model';

import { RecordService } from '../shared/service/record.service';

import { Observable, Subscription } from 'rxjs/Rx';

1. Add the following to the signature of the component class:-

public records: Observable<Array<Record>>;

public model: Record;

constructor(

private route: ActivatedRoute,

private recordService: RecordService

) { }

ngOnInit(): void {

// obtain dataset

this.records = this.route.snapshot.data['record'];

this.model = new Record({});

}

1. Add the following code method:-

saveRecord(record: Record): Observable<Array<Record>> {

if (record.\_id == null) {

this.recordService.createRecord(record).subscribe(record => this.records = this.getAllRecords());

} else {

this.recordService.updateRecord(record).subscribe(record => this.records = this.getAllRecords());

}

return this.records;

}

1. Add the onSubmit() method:-

onSubmit(newForm: NgForm): void {

//

this.submitted = true;

this.currentRecord = JSON.stringify(this.model);

//submit new record to database

this.saveRecord(this.model);

//

newForm.reset();

}

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the following code method:-

<form #newForm="ngForm" (ngSubmit)="onSubmit(newForm)" novalidate>

<div class="form-group">

<label for="firstName">First Name</label>

<input type="text" class="form-control" id="firstName" minlength="3" maxlength="30" placeholder="Please enter the first name"

name="firstName" [(ngModel)]="model.firstName" #firstName="ngModel" required>

<div \*ngIf="firstName.errors && (firstName.dirty || firstName.touched)" class="alert alert-danger">

<div [hidden]="!firstName.errors.required">A name is required!</div>

<div [hidden]="!firstName.errors.minlength">the name must be at least 3 characters long.</div>

<div [hidden]="!firstName.errors.maxlength">the name cannot be more than 30 characters long.</div>

</div>

</div>

<div class="form-group">

<label for="lastName">Last Name</label>

<input type="text" class="form-control" id="lastName" minlength="3" maxlength="30" placeholder="Please enter the last name"

name="lastName" [(ngModel)]="model.lastName" #lastName="ngModel" required>

<div \*ngIf="lastName.errors && (lastName.dirty || lastName.touched)" class="alert alert-danger">

<div [hidden]="!lastName.errors.required">A name is required!</div>

<div [hidden]="!lastName.errors.minlength">the name must be at least 3 characters long.</div>

<div [hidden]="!lastName.errors.maxlength">the name cannot be more than 30 characters long.</div>

</div>

</div>

<div class="form-group">

<label for="emailAddress">Email Address</label>

<input type="text" class="form-control" id="emailAddress" placeholder="Please enter an email address" name="emailAddress"

[(ngModel)]="model.emailAddress" #emailAddress="ngModel" required>

<div \*ngIf="emailAddress.errors && (emailAddress.dirty || emailAddress.touched)" class="alert alert-danger">

<div [hidden]="!emailAddress.errors.required">An email address is required!</div>

</div>

</div>

<button type="submit" class="\_btn \_btn-success" mat-raised-button color="accent" [disabled]="!newForm.form.valid">Submit</button>

<button type="button" class="\_btn \_btn-default" mat-raised-button color="primary" (click)="newForm.reset()">Clear</button>

</form>

## Module Imports

1. Open **app.module.ts** in the following location:-

**<proj\_root>/webapp/app /app.module.ts**

1. Enter the following code imports:-

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

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

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

import { HttpClientModule } from "@angular/common/http";

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

import { PortalComponent } from './portal/portal.component';

import { AppRoutingModule } from "./app-routing/app-routing.module";

import { RecordResolve } from "./shared/resolve/record-resolve.service";

import { RecordService } from "./shared/service/record.service";

1. Locate the @NgModule section of the file and insert the following code:-

@NgModule({

declarations: [

AppComponent,

PortalComponent,

],

imports: [

BrowserModule,

AppRoutingModule,

HttpModule,

HttpClientModule

],

providers: [

RecordService,

RecordResolve

],

bootstrap: [AppComponent]

})

## Routing

1. Create/Open **app-routing.module.ts** in the following location:-

**<proj\_root>/webapp/app/ app-routing / app-routing -routing.module.ts**

1. Open the file and enter the following code imports:-

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

import { RouterModule, Routes } from '@angular/router';

// Component

import { PortalComponent } from '../portal/portal.component'

// Resolve

import { RecordResolve } from '../shared/resolve/record-resolve.service';

1. Locate the routes[] section of the file and insert the following code:-

const routes: Routes = [

{

path: '', component: PortalComponent, resolve: { record: RecordResolve }

},

];

# Create Form

This section will detail how a form is implement using Angular 2. The following instruction will include the code required to consume the model and form validation.

## Pre-requisites

1. Open **package.json** in the following location:-

**<proj\_root>/package.json**

1. Ensure the **@angular/forms** dependency is included:-

"@angular/forms": "^5.0.0",

1. Open **app.module.ts** from the following location:-

**<proj\_root>/src/app/app.module.ts**

1. Add the following import:-

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

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the **NgForm** import:-

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

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the following **<form>** element:-

<form #newForm="ngForm" novalidate>

<div class="form-group">

<label for="firstName">First Name</label>

<input type="text" class="form-control" id="firstName" placeholder="Please enter the first name"

name="firstName">

</div>

<div class="form-group">

<label for="lastName">Last Name</label>

<input type="text" class="form-control" id="lastName" placeholder="Please enter the last name"

name="lastName">

</div>

<div class="form-group">

<label for="emailAddress">Email Address</label>

<input type="text" class="form-control" id="emailAddress" placeholder="Please enter the email address"

name="emailAddress">

</div>

<button type="submit" class="btn btn-success">Submit</button>

</form>

1. Open the project in the browser to ensure the form elements are displayed.

## Add Model

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the **Record** model from the Angular core component:-

import { Record } from '../core/record.model';

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the respective model attribute to the corresponding **<input>** element:-

e.g.

<input type="text" … [(ngModel)]="model.firstName" #firstName="ngModel">

## Add Validation

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the respective model attribute to the corresponding **<input>** element:-

eg.

<input … minlength="3" maxlength="30">

1. Add a **<div>** tag section beneath each **<input>** tag corresponding to the firstName and **lastName** elements and insert the following code:-

eg.

<div \*ngIf="lastName.errors && (firstName.dirty || firstName.touched)" class="alert alert-danger">

<div [hidden]="!firstName.errors.required">A name is required!</div>

<div [hidden]="!firstName.errors.minlength">the name must be at least 3 characters long.</div>

<div [hidden]="!firstName.errors.maxlength">the name cannot be more than 30 characters long.</div>

</div>

1. Add the following attribute to the submit button:-

<button type="button" … [disabled]="!newForm.form.valid">

1. Add a button to reset the form:-

<button type="button" class="\_btn \_btn-default" mat-raised-button color="primary" (click)="newForm.reset()">Clear</button>

# Display Record

The following section details the functionality behind displaying the record to screen. When a user clicks on a record listing, the record details will populate the form.

## Create API Service

1. Open **portalService.js** from the following location:-

**<proj\_root>/server/src/portal/portalService.js**

1. Add the following methods to the component class:-

exports.findRecordById = (id) => Record.findById(id)

.exec();

1. Open **portalController.js** from the following location:-

**<proj\_root>/server/src/portal/ portalController.js**

1. Add the following methods to the component class:-

exports.findRecordById = function (req, res) {

var id = req.params ? req.params.id : undefined;

if (!id) {

res.status(400).send("Retrieve Operation Error: Invalid Identifier");

} else {

var onErr = (err) => {

res.status(500).send(err);

};

var onSuccess = (data) => {

var response = res.status(200);

};

portalService.findRecordById(id).then(data => res.status(200).send(data));

}

}

1. Open **portalRoutes.js** from the following location:-

**<proj\_root>/server/src/portal/ portalRoutes.js**

1. Add the following route:-

router.get('/:id', portalController.findRecordById);

## Create Angular 2 Shared Service

1. Open **record.service.ts** from the following location:-

**<proj\_root>/app/shared/service/record.service.ts**

1. Add the following method:-

getRecordById(id: string): Observable<Record> {

let headers = new Headers();

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

let options = new RequestOptions({ headers: headers });

return this.http.get('/api/portal/' + id, options)

.map((res: Response) => new Record(res.json()))

.catch(error => Observable.throw(error.json().error || 'Server error'))

}

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the following method to the component class:-

getRecordById(id: string): Record {

this.recordService.getRecordById(id).subscribe(record => this.model = new Record(record));

return this.model;

}

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the following **click()** attribute to the **NgFor()** **<li>** tag:-

(click)="getRecordById(record.\_id)"

# Search Record

## Create API Service

1. Open **portalService.js** from the following location:-

**<proj\_root>/server/src/portal/portalService.js**

1. Add the following code for the **findRecordByQuery()** function:-

exports.findRecordByQuery = (search) => {

var query = {};

if (search != null) {

let rxSearch = new RegExp(search.toLowerCase().trim(), 'i')

query = {

$or: [

{ 'firstName': rxSearch },

{ 'lastName': rxSearch },

{ 'emailAddress': rxSearch }

]

}

}

return Record.find(query)

};

1. Open **portalController.js** from the following location:-

**<proj\_root>/server/src/portal/portalController.js**

1. Add the following code for the **findRecordByQuery()** function:-

exports.findRecordByQuery = function (req, res) {

var query = req.query ? req.query.query : undefined;

if (!query) {

res.status(400).send("Query Operation Error: Invalid parameter(s)");

} else {

var onErr = (err) => {

res.status(500).send(err);

};

var onSuccess = (data) => {

var response = res.status(200);

};

portalService.findRecordByQuery(query)

.then(data => res.status(200).send(data));

}

}

1. Open **portalRoutes.js** from the following location:-

**<proj\_root>/server/src/portal/portalRoutes.js**

1. Add the following route:-

router.get('/search/:param', portalController.findRecordByQuery);

## Create Angular 2 Shared Service

1. Open **record.service.ts** from the following location:-

**<proj\_root>/app/shared/service/record.service.ts**

1. Ensure the following has been included in **@angular/http** import:-

import { ..., URLSearchParams } from '@angular/http';

1. Add the following method:-

getRecordBySearchParam(search: string): Observable<any> {

let headers = new Headers();

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

let params: URLSearchParams = new URLSearchParams();

if (search) {

params.set('query', search);

}

let options = new RequestOptions({ headers: headers, search: params });

return this.http.get('/api/portal/search/' + search, options)

.map(response => (response.json() as Object[]).map(record => new Record(record)))

.catch(error => Observable.throw(error.json().error || 'Server error'))

}

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the following method to the component class:-

getRecordBySearchParam(search: string): Observable<Array<Record>> {

this.recordService.getRecordBySearchParam(search).subscribe(record => this.records = record);

return this.records;

}

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the following **<form>** tag:-

<form #search="ngForm" novalidate>

<div class="form-group">

<label for="searchParam" class="h4 col-3">Search</label>

<input type="text" class="form-control" id="searchParam" name="searchParam" placeholder="search" [(ngModel)]="searchParam"

(ngModelChange)="getRecordBySearchParam($event)">

</div>

</form>

# Angular Material

## Introduction

This project uses Angular Material to stylize the HTML component. The following instruction is a guide on how to install and implement this feature.

## Installation

1. Open the Command Prompt at the project root.
2. To install Angular Material and Angular Animations execute the following at the Command Prompt:-

npm install --save @angular/material @angular/animations @angular/cdk

1. Restart the server for the changes to take effect.

## Implementation

1. Create **portal.module.ts** in the following location:-

**<proj\_root>/webapp/app/portal/ portal.module.ts**

1. Ensure the following imports are present:-

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

import { MatButtonModule, MatDialogModule } from '@angular/material';

1. Ensure the following declarations have been made:-

@NgModule({

  imports: [

    MatButtonModule,

MatDialogModule

  ],

  exports: [

    MatButtonModule,

    MatDialogModule

  ]

})

1. Open **app.module.ts** from the following location:-

**<proj\_root>/webapp/app/app.module.ts**

1. Ensure the following imports are present:-

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

1. Ensure the following declarations have been made:-

imports [

…

    BrowserAnimationsModule,

…

],

1. Open **style.css** from the following location:-

**<proj\_root>/styles.css**

1. Ensure the following imports are present:-

@import '~https://fonts.googleapis.com/icon?family=Material+Icons';

@import '~@angular/material/prebuilt-themes/indigo-pink.css';

# Delete Record functionality

1. Open **portal.module.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Ensure the following declaration is present in **imports[]** section of **@NgModule**:-

@NgModule({

  imports: [

    ...

MatDialogModule

  ],

1. Open the Command Prompt at the project rootand executethe following command where **<name>** is the name of the component.

ng g component <name>

For the purposes of this example, name this component **delete-dialog**.

1. Open **delete-dialog.component.ts** from the following location:-

**<proj\_root>/webapp/app/ delete-dialog/ delete-dialog.component.ts**

1. Enter the following code imports:-

import { Component, Inject, OnInit } from '@angular/core';

import { MAT\_DIALOG\_DATA, MatDialog } from '@angular/material';

import { RecordService } from '../shared/service/record.service';

1. Insert the following injections into the class **constructor()** signature:-

constructor(

@Inject(MAT\_DIALOG\_DATA)

public data: any,

private recordService: RecordService,

private dialog: MatDialog

) { }

1. Add the following methods to the component class:-

dialogClose() {

this.dialog.closeAll();

}

deleteRecord(id: string): void {

this.dialogClose();

}

1. Subscribe to the **deleteRecord()** service call in the following method:-

deleteRecord(id: string): void {

this.recordService.deleteRecord(id).subscribe();

this.dialogClose();

}

1. Open **delete-dialog.component.html** from the following location:-

**<proj\_root>/webapp/app/ delete-dialog/ delete-dialog.component.html**

1. Enter the following code to render the dialog box:-

<h2 mat-dialog-title>Delete Record</h2>

<span mat-dialog-content>Are you sure?</span>

<span mat-dialog-actions>

<button mat-button mat-dialog-close (click)="dialogClose();">No</button>

<button (click)="deleteRecord(data.id);">Yes</button>

</span>

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/ portal/ portal.component.ts**

1. Enter the following code imports:-

import { MatDialog, MatDialogRef } from '@angular/material';

import { DeleteDialogComponent } from '../delete-dialog/delete-dialog.component';

1. Add the following methods to the component class:-

openAreYouSureDialog(id: string): void {

let dialog = this.dialog.open(DeleteDialogComponent, {

data: { id: id }

});

dialog.afterClosed()

.subscribe(record => this.records = this.getAllRecords());

}

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/ portal/ portal.component.html**

1. Insert the following code inside the **NgFor()** <**div>** tag:-

<span>

<mat-icon color="primary" aria-label="Delete">delete</mat-icon>

</span>

1. Add the following click event to the mat-icon<> element:-

(click)="openAreYouSureDialog(record.\_id)"

# File Upload

## Introduction

A file upload feature will be implemented to upload new records into the database. The user will be able to browse the file system for the compatible .csv file and click the upload button which will first upload the file to the server and then import its contents to the application database. The imported data will be instantaneously displayed to the user.

This section will focus on the file upload implementation, ahead of the next section which will complete the CSV import functionality encapsulated on the upload button of the form.

## Installation

1. Install the **express-fileupload** npm module:-

npm install express-fileupload -–save

1. Install the **ng2-file-upload** npm module:-

npm install ng2-file-upload –-save

1. Create a folder named upload at the following path location:-

**<proj\_root>/server/upload/**

This folder will serve as the location where the uploaded CSVs are stored.

## Implementation

1. Open **server.js** from the following location:-

**<proj\_root>/server.js**

1. Add the following import:-

const fileUpload = require('express-fileupload');

1. Add the following import:-

add.use(fileUpload());

1. Open **portalController.js** from the following location:-

**<proj\_root>/server/src/portal/ portalController.js**

1. Add the following function to the class:-

exports.uploadCsv = function (req, res) {

var file = req.files.csvfile;

var filepath = path.join(appRoot, 'server/upload/' + file.name);

file.mv(filepath, function (err) {

if (err) {

res.send('error occurred');

} else {

res.send('Done!');

}

})

};

1. Open **portalRoutes.js** from the following location:-

**<proj\_root>/server/src/portal/ portalRoutes.js**

1. Add the following route:-

router.post('/file/upload', portalController.uploadCsv);

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the following import to the comonent:-

import { FileUploader } from 'ng2-file-upload/ng2-file-upload';

1. Add the following declaration before the **@component()** code section:-

const URL = 'http://localhost:3000/api/portal/file/upload';

1. Add the following declaration in the component class:-

public uploader: FileUploader = new FileUploader({ url: URL, itemAlias: 'csvfile' });

1. Add the following code in the **OnInit()** code method:-

this.uploader.onAfterAddingFile = (file) => { file.withCredentials = false; };

this.uploader.onCompleteItem = (item: any, response: any, status: any, headers: any) => {

// execution code on upload completion

};

**Note:** When the CSV import method is implemented this will replace the above comment:-

1. Open **portal.component.html** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.html**

1. Add the following **form<>** element:-

<form #upload="ngForm" novalidate>

<div class="form-group">

<label for="file">Upload File</label>

<input type="file" class="form-control" name="csvfile" accept=".csv" ng2FileSelect [uploader]="uploader" />

</div>

<button type="button" class="btn btn-success btn-s" (click)="uploader.uploadAll()" [disabled]="!uploader.getNotUploadedItems().length">Upload Contacts</button>

</form>

# CSV Import

## Introduction

This section will implement the data import routine. This will complete the expected functionality of the file upload feature, enabling the user to import the csv file as part of a one-step process.

## Install

1. Install the **csvtojson** npm module:-

npm install csvtojson –-save

## Implementation

1. Open **portalController.js** from the following location:-

**<proj\_root>/server/src/portal/ portalController.js**

1. Add the following function to the class:-

exports.importCsv = function (req, res) {

console.log('portalController: importCsv');

var filepath = path.join(appRoot, 'server/upload/');

var csvFilePath = path.join(filepath, 'test.csv');

csvtojson()

.fromFile(csvFilePath)

.on('json', (jsonObj) => {

//

portalService.createRecord(jsonObj)

.then(data => res.status(200).send(data))

.catch('an error occurred');

})

.on('done', (error) => {

console.log('end')

})

};

1. Open **portal.component.ts** from the following location:-

**<proj\_root>/webapp/app/portal/ portal.component.ts**

1. Add the following subscription to the **onCompleteItem** method:-

this.uploader.onCompleteItem = (item: any, response: any, status: any, headers: any) => {

this.recordService.importCsv().subscribe(record => this.records = this.getAllRecords());

};

1. Run the project in the browser and upload a CSV file.