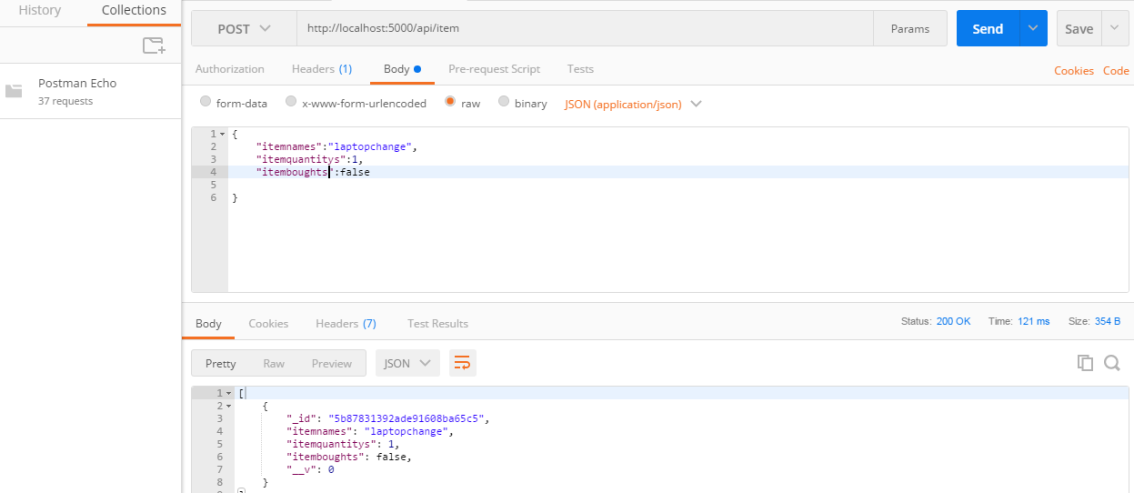
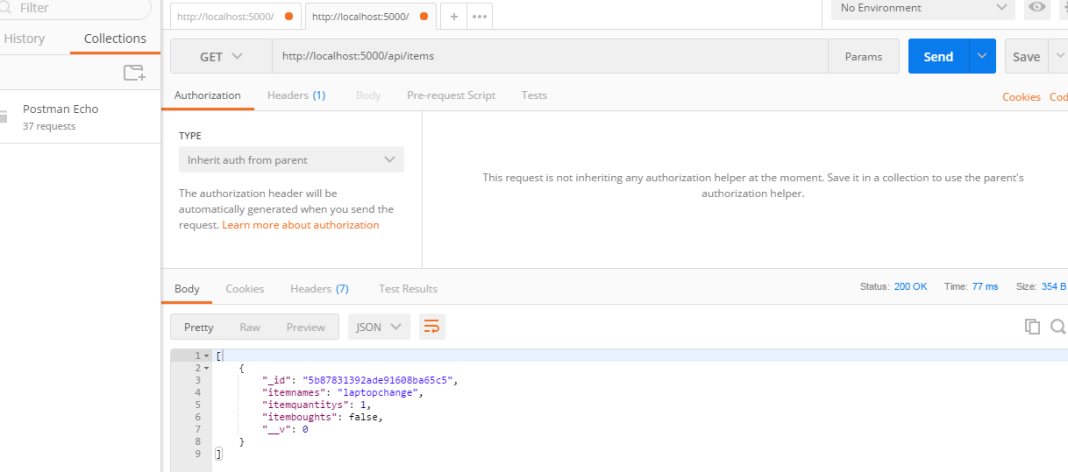
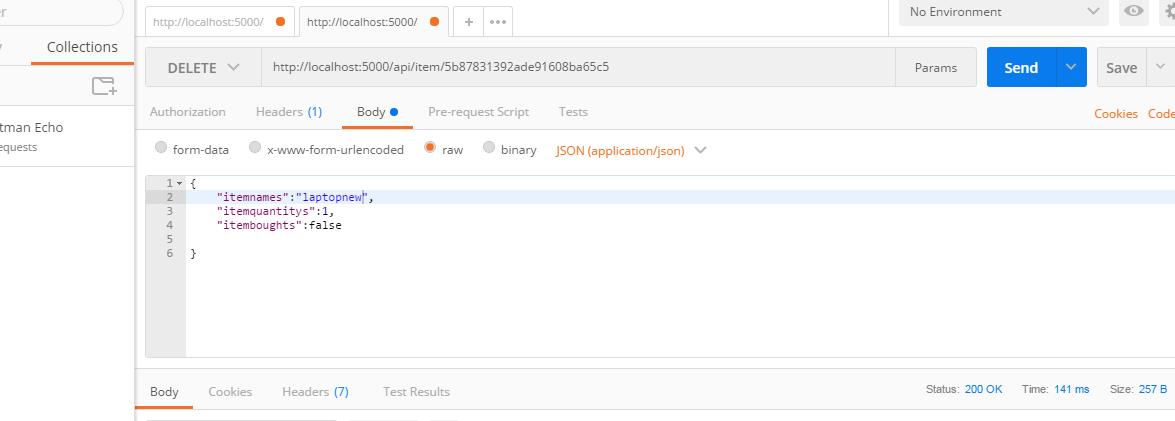
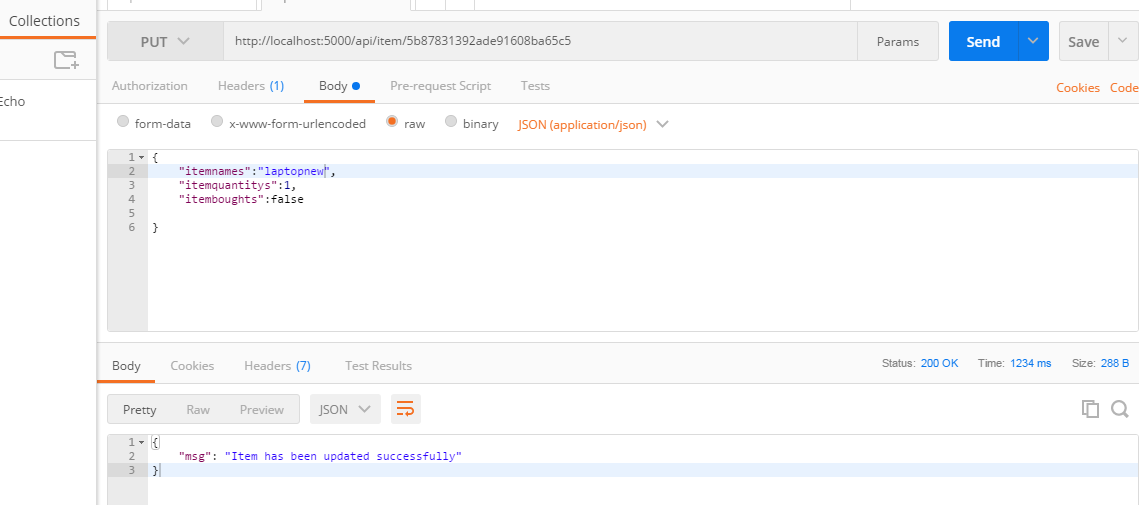
Mean Stack Shopping cart application-MEAN Integration

1. Create an front end project using angular
2. Ng new shoppingcartFE
3. Create an back end project using Node and Express
4. First install nodejs
5. Install mongodb
6. Create an project for Node and express
7. Create an folder shoppingcartBE
8. Npm init –this will create an package.json file
9. Install the middle wares like mongoose and express
10. Npm install express --save
11. Npm install mongoose --save
12. Npm install cors --save
13. Npm install body-parser –save
14. For database mongodb you need to start the mongod process
15. Open an command prompt and type
16. Mongod –dbpath “E:\Search-Data\Syntel Batch\Syntel\Syntel\_TR\_Data\_2\Syntel\_TR\_Data\_2\8. MEAN STACK\syntel\_Mean\_case\_study\_3days\data\db”
17. You can open second command prompt for mongodb client
18. Just give mongo command on the command prompt
19. In Syntel we can create an folder shoppingcart inside that copy angular5-example and expressfolder from common drive
20. Open the shoppingcart folder in visual studio code
21. Open termimal and go to shoppingcartBE
22. Create package.json
23. Npm init
24. Continue as step 10, 11 and 12, 13
25. For backend
26. Create model and route folder
27. Now create an entry.js from this file the program starts executing
28. Create an items.js file for model folder
29. Create an routes.js file inside routes folder
30. To run the backend project
31. Node entry.js
32. It will start the node server but u need to start mongodb server as step no 16
33. To test your express and node service you can use postman
34. And u can perform the get and post methods
35. The code for all three files as as bellow
36. Code for items.js for model
37. const mongoose = require('mongoose');
38. const itemsschema = mongoose.Schema({
39. itemnames: {
40. type: String,
41. required: false,
42. },
43. itemquantitys: {
44. type: Number,
45. required: false,
46. },
47. itemboughts: {
48. type: Boolean,
49. required: false,
50. },
51. });
52. const items=module.exports=mongoose.model('items',itemsschema);
53. code for routes.js
54. var express = require('express');
55. var router = express.Router();
56. const Item = require('../model/items');
57. router.get('/', (req, res) => {
58. console.log('hello route');
59. });
60. router.get('/items', (req, res, next) => {
61. Item.find(function (err, items) {
62. if (err) {
63. res.json(err);
65. }
66. else {
67. res.json(items);
68. }
69. });
70. });
71. router.post('/item',(req,res,next)=>{
72. const newitem=new Item(
73. {
74. itemnames:req.body.itemnames,
75. itemquantitys:req.body.itemquantitys,
76. itemboughts:req.body.itemboughts
77. });
78. newitem.save((err,item)=>{
79. if(err)
80. {
81. res.json(err);
82. }
83. else
84. {
85. res.json({msg:'Item has been added successfully'});
86. }
87. });
88. });
90. router.delete('/item/:id', (req, res, next) => {
91. //deleting data
92. Item.remove({ \_id: req.params.id }, function (err, result) {
93. if (err) {
94. res.json(err);
95. }
96. else {
97. res.json(result);
98. }
99. });
100. });
101. router.put('/item/:id', (req, res, next) => {
102. //updating data
103. Item.findOneAndUpdate({ \_id: req.params.id }, {
104. $set: {
105. itemnames: req.body.itemnames,
106. itemquantitys: req.body.itemquantitys,
107. itemboughts: req.body.itemboughts
108. }
109. }, function (err, result) {
110. if (err) {
111. res.json(err);
112. }
113. else {
114. // res.json(result);
115. res.json({ msg: 'Item has been updated successfully' })
116. }
117. })
118. });
119. module.exports=router;
120. Code for entry.js
121. var express = require('express');
122. var mongoose = require('mongoose');
123. var bodyparser = require('body-parser');
124. var cors = require('cors');
125. const route = require('./route/routes');
126. var app = express();
127. //connect to mangodb
128. mongoose.connect('mongodb://localhost:27017/shoppinglistsyntel', {useNewUrlParser: true});
129. //on connetion
130. mongoose.connection.on('connected', () => {
131. console.log('Mongodb connected at post 27017');
132. });
133. mongoose.connection.on('error', (err) => {
134. console.log(err);
135. });
136. //adding middle ware -cors
137. app.use(cors());
138. app.use(bodyparser.json());
139. app.use('/api', route);
140. const PORT=5000
141. app.listen(PORT, function () {
142. console.log("Server Started….." +PORT);
143. })







1. Now the back end application is ready now we will create front end application
2. Go to index .html and comment all code and write nav bar using bootstrap and add the router outlet code
3. AppModule we have give route array
4. const appRoutes: Routes = [
5. { path: 'shop', component: ShoppingcartitemComponent }
7. ];
8. Path or library
9. import { RouterModule, Routes } from '@angular/router';
10. import { FormsModule } from '@angular/forms';
11. imports: [
12. BrowserModule,
13. FormsModule,//this this forms module added
14. RouterModule.forRoot(appRoutes)
15. ],

Code for app.component.html

<h1>Shopping cart Application</h1>

<div class="container">

<div class="row">

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container">

<a class="navbar-brand">Shopping Cart</a>

<ul class="nav navbar-nav" routerLinkActive="active">

<li class="nav-item">

<a class="nav-link" routerLink="shop">Shopping Item </a>

</li>

</ul>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-md-8">

<router-outlet></router-outlet>

</div>

</div>

</div>

<nav class="navbar navbar-light bg-primary" role="navigation">

<div class="container">

<b>Happy Learning Angular </b>

</div>

</nav>

</div>

</div>

Create an component ng g c ShoppingcartitemComponent

Code for ShoppingcartitemComponent html page

<form #frm="ngForm" (ngSubmit)="addItem(frm)" \*ngIf="!toggleForm">

<h2>Add Item</h2>

<div class="form-group">

<label for="">

Name:

</label>

<input type="text" class="form-control" required name="itemnames" value="" ngModel>

</div>

<div class="form-group">

<label for="">

Quantity:

</label>

<input type="number" class="form-control" required name="itemquantitys" value="" ngModel>

</div>

<button type="submit" name="button" class="btn btn-success" [disabled]=frm.invalid>Add Item</button>

</form>

<form #editfrm="ngForm" (ngSubmit)="editForm(editfrm)" \*ngIf="toggleForm">

<h2>Edit Item</h2>

<div class="form-group">

<label for="">

Name:

</label>

<input type="text" class="form-control" name="itemnames" value="" [ngModel]="selectedItem.itemnames">

</div>

<div class="form-group">

<label for="">

Quantity:

</label>

<input type="number" class="form-control" name="itemquantitys" value="" [ngModel]="selectedItem.itemquantitys">

</div>

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

</form>

<h2>Shopping List </h2>

<div class="row">

<div \*ngFor="let item of shoppinglistitem" class="">

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

<input type="checkbox" name=""

[checked]="item.itemBought"

(click)='updateItemcheckbox(item)'>

</div>

<div class="col-md-3">

{{item.itemnames}}

</div>

<div class="col-md-2">

{{item.itemquantitys}}

</div>

<div class="col-md-3">

<button type="button" name="button" class="btn btn-primary" (click)='showEditform(item)' [disabled]='toggleForm'>Edit</button>

</div>

<div class="col-md-3">

<button type="button" name="button" class="btn btn-danger" (click)='deleteItem(item.\_id)'>Delete</button>

<br><br>

</div>

</div>

Code for ShoppingcartitemComponent.ts file

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

import { Item } from '../model/item';

import { DataService } from '../services/data.service';

@Component({

selector: 'app-shoppingcartitem',

templateUrl: './shoppingcartitem.component.html',

styleUrls: ['./shoppingcartitem.component.css']

})

export class ShoppingcartitemComponent implements OnInit {

shoppinglistitem: Item[] = [];

toggleForm: Boolean = false;

selectedItem: Item;

constructor(private dataservice: DataService) { }

ngOnInit() {

this.getItems();

}

getItems() {

//to do later

this.dataservice.getshoppingitems()

.subscribe(items => {

this.shoppinglistitem = <Item[]>items;

// console.log('data from data service:'+this.shoppinglistitem[0].itemName + " " +this.shoppinglistitem[1].itemQuantity );

})

}

addItem(frm) {

console.log(frm.value);

let newitem: Item = {

itemnames: frm.value.itemnames,

itemquantitys: frm.value.itemquantitys,

itemboughts: false

}

this.dataservice.addshoppingitem(newitem)

.subscribe(item => {

console.log(item);

this.getItems();

})}

editForm(editfrm) {

let newItem: Item = {

\_id: this.selectedItem.\_id,

itemnames: editfrm.value.itemnames,

itemquantitys: editfrm.value.itemquantitys,

itemboughts: this.selectedItem.itemboughts

}

this.dataservice.updateshoppingitem(newItem)

.subscribe(result => {

console.log('original item to be updated with old values' + result);

this.getItems();

this.toggleForm = !this.toggleForm;

})

}

showEditform(Item) {

this.selectedItem = Item;

this.toggleForm = !this.toggleForm;

}

updateItemcheckbox(item) {

item.itemboughts = !item.itemboughts;

this.dataservice.updateshoppingitem(item)

.subscribe(result => {

// console.log('Original CheckBox values' + result.itemboughts);

// item.getItems();

})

}

deleteItem(id) {

this.dataservice.deleteshoppingItem(id)

.subscribe(result => {

console.log(result);

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

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

this.shoppinglistitem.splice(i, 1);

}

}

})

}

}

1. Create an model folder and create item.ts file
2. export class Item{
3. \_id?:string;
4. itemnames:string;
5. itemquantitys:Number;
6. itemboughts:Boolean
7. }
8. Create an Services folder and create data.service.ts file and the code is as bellow
9. import { Injectable } from '@angular/core';
10. import { HttpClient } from '@angular/common/http';
11. @Injectable()
12. export class DataService {
13. constructor(private http: HttpClient) {
14. }
15. getshoppingitems() {
16. return this.http.get('http://localhost:5000/api/items');
17. }
18. addshoppingitem(additem) {
19. return this.http.post('http://localhost:5000/api/item', additem);
20. }
21. deleteshoppingItem(id) {
22. return this.http.delete('http://localhost:5000/api/item/' + id)
23. }
24. updateshoppingitem(newItem)
25. {
26. return this.http.put('http://localhost:5000/api/item/'+newItem.\_id,newItem);
28. }
29. }

Code for AppModule file

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

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

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

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

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

import { ShoppingcartitemComponent } from './shoppingcartitem/shoppingcartitem.component';

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

import { DataService } from './services/data.service';

const appRoutes: Routes = [

{ path: 'shop', component: ShoppingcartitemComponent }

];

@NgModule({

declarations: [

AppComponent,

ShoppingcartitemComponent

],

imports: [

BrowserModule,

HttpClientModule,//this is httpclientmodule added

FormsModule,//this this forms module added

RouterModule.forRoot(appRoutes)

],

providers: [DataService],

bootstrap: [AppComponent]

})

export class AppModule { }