Lappeenrannan teknillinen yliopisto

School of Software Engineering

Software Development Skills: Full-Stack

Naosad Hossen, 001489455

LEARNING DIARY, FULL-STACK 2023-24 MODULE

**LEARNING DIARY**

**28.12.2023**

I enrolled to “Software Development Skill: Full-Stack 2023-24” to learn and understand how software development works in this kind of technology. As a software project manager, it is important to understand this popular technology. So, I am excited to participate in this course and gaining practical knowledge about Full-Stack technology.

I went through “General Course Information” and understood the course goal is to improve my competitiveness in the job market by providing tools for creating unique projects and to help me find your passion as a software developer.

I will reuse the code provided in the webcourse and extend it with a simple feature to manage customer contact information to demonstrate my understanding on how MEAN Stack works.

# Watching the Videos

## Watching the “INTRO TO VERSION CONTROL” video.

Refreshed my memory on Git Basic Commands:

Working with Local repo: init, add, status, commit,

Working with Remote repo: push, pull, clone

Agree with the author that we should use command as much as possible to understand how Git works under the hood and to survive if that tool was not available. Author also suggested to use Git Bash in Windows over CMD command line tool.

## Watching the “NodeJS” video

Here I learnt:

how to install NodeJS (go to Nodejs.org and download the installation file)

How to initialize a NodeJS project (npm init)

How to install package globally (sudo npm install -g <package name> )

How to execute a NodeJS project using script (npm start)

How to install nodemon as dev dependency (npm install -D nodemon)

How to execute a NodeJS project with nodemon to automatically load the program after any change in the code (simply run nodemon from project directory)

During the coding, I learnt how to use http, path, fs, url, event and os module to build a http server and do file system operation in NodeJS.

Codes are here: <https://github.com/naosadhossen/nodejs>

## Watching the “MongoDB” video

In this course, I learnt how to install MongoDB in MacOS using homebrew. After installing, used mongo shell CLI to connect to database. Also installed Mongo Compass GUI and connected to the database. Ran the mondo shell commands provided in the course.

naosadhossen@Naosad-Hossen bin % sudo ./mongo

Password:

MongoDB shell version v3.6.23

connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("542b9ce4-1ea1-4c7a-904e-a8319a52a6bf") }

MongoDB server version: 3.6.23

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

http://docs.mongodb.org/

Questions? Try the support group

http://groups.google.com/group/mongodb-user

Server has startup warnings:

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten]

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* WARNING: Access control is not enabled for the database.

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* Read and write access to data and configuration is unrestricted.

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten]

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* WARNING: This server is bound to localhost.

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* Remote systems will be unable to connect to this server.

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* Start the server with --bind\_ip <address> to specify which IP

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* addresses it should serve responses from, or with --bind\_ip\_all to

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* bind to all interfaces. If this behavior is desired, start the

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten] \*\* server with --bind\_ip 127.0.0.1 to disable this warning.

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten]

2023-12-28T13:22:21.122+0200 I CONTROL [initandlisten]

2023-12-28T13:22:21.123+0200 I CONTROL [initandlisten] \*\* WARNING: soft rlimits too low. Number of files is 256, should be at least 1000

**> show dbs**

admin 0.000GB

config 0.000GB

customerdb 0.000GB

local 0.000GB

**> db**

test

**> use acme**

switched to db acme

**> db.dropDatabase()**

{ "ok" : 1 }

**> db.createCollection('posts')**

{ "ok" : 1 }

**> show collections**

posts

**> db.posts.insert**({

... title: 'Post One',

... body: 'Body of post one',

... category: 'News',

... tags: ['news', 'events'],

... user: {

... name: 'John Doe',

... status: 'author'

... },

... date: Date()

... })

WriteResult({ "nInserted" : 1 })

> **db.posts.insertMan**y([

... {

... title: 'Post Two',

... body: 'Body of post two',

... category: 'Technology',

... date: Date()

... },

... {

... title: 'Post Three',

... body: 'Body of post three',

... category: 'News',

... date: Date()

... },

... {

... title: 'Post Four',

... body: 'Body of post three',

... category: 'Entertainment',

... date: Date()

... }

... ])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("658eb7f21a0ffa25d40ff2d2"),

ObjectId("658eb7f21a0ffa25d40ff2d3"),

ObjectId("658eb7f21a0ffa25d40ff2d4")

]

}

**> db.posts.find()**

{ "\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"), "title" : "Post One", "body" : "Body of post one", "category" : "News", "tags" : [ "news", "events" ], "user" : { "name" : "John Doe", "status" : "author" }, "date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)" }

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"), "title" : "Post Two", "body" : "Body of post two", "category" : "Technology", "date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)" }

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d3"), "title" : "Post Three", "body" : "Body of post three", "category" : "News", "date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)" }

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d4"), "title" : "Post Four", "body" : "Body of post three", "category" : "Entertainment", "date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)" }

**> db.posts.find().pretty()**

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"),

"title" : "Post Two",

"body" : "Body of post two",

"category" : "Technology",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d3"),

"title" : "Post Three",

"body" : "Body of post three",

"category" : "News",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d4"),

"title" : "Post Four",

"body" : "Body of post three",

"category" : "Entertainment",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

**> db.posts.find**({ category: 'News' })

{ "\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"), "title" : "Post One", "body" : "Body of post one", "category" : "News", "tags" : [ "news", "events" ], "user" : { "name" : "John Doe", "status" : "author" }, "date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)" }

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d3"), "title" : "Post Three", "body" : "Body of post three", "category" : "News", "date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)" }

> # asc

2023-12-29T14:14:10.595+0200 E QUERY [thread1] SyntaxError: illegal character @(shell):1:0

**> db.posts.find().sort**({ title: 1 }).pretty()

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d4"),

"title" : "Post Four",

"body" : "Body of post three",

"category" : "Entertainment",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d3"),

"title" : "Post Three",

"body" : "Body of post three",

"category" : "News",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"),

"title" : "Post Two",

"body" : "Body of post two",

"category" : "Technology",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

> # desc

2023-12-29T14:14:10.600+0200 E QUERY [thread1] SyntaxError: illegal character @(shell):1:0

**> db.posts.find().sort**({ title: -1 }).pretty()

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"),

"title" : "Post Two",

"body" : "Body of post two",

"category" : "Technology",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d3"),

"title" : "Post Three",

"body" : "Body of post three",

"category" : "News",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d4"),

"title" : "Post Four",

"body" : "Body of post three",

"category" : "Entertainment",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

**> db.posts.find().count()**

4

**> db.posts.find({ category: 'news' }).count()**

0

**> db.posts.find().limit(2).pretty()**

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"),

"title" : "Post Two",

"body" : "Body of post two",

"category" : "Technology",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

> db.posts.find().limit(2).sort({ title: 1 }).pretty()

{

"\_id" : ObjectId("658eb7f21a0ffa25d40ff2d4"),

"title" : "Post Four",

"body" : "Body of post three",

"category" : "Entertainment",

"date" : "Fri Dec 29 2023 14:13:38 GMT+0200 (EET)"

}

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

**> db.posts.find().forEach(function(doc)** {

... print("Blog Post: " + doc.title)

... })

Blog Post: Post One

Blog Post: Post Two

Blog Post: Post Three

Blog Post: Post Four

**> db.posts.findOne**({ category: 'News' })

{

"\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"),

"title" : "Post One",

"body" : "Body of post one",

"category" : "News",

"tags" : [

"news",

"events"

],

"user" : {

"name" : "John Doe",

"status" : "author"

},

"date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)"

}

**> db.posts.find**({ title: 'Post One' }, {

... title: 1,

... author: 1

... })

{ "\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"), "title" : "Post One" }

**> db.posts.update({** title: 'Post Two' },

... {

... title: 'Post Two',

... body: 'New body for post 2',

... date: Date()

... },

... {

... upsert: true

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.posts.updat**e({ title: 'Post Two' },

... {

... $set: {

... body: 'Body for post 2',

... category: 'Technology'

... }

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.posts.update**({ title: 'Post Two' },

... {

... $inc: {

... likes: 5

... }

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.posts.update**({ title: 'Post Two' },

... {

... $rename: {

... likes: 'views'

... }

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

**> db.posts.update**({ title: 'Post Two' },

... {

... $rename: {

... likes: 'views'

... }

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

**> db.posts.remove**({ title: 'Post Four' })

WriteResult({ "nRemoved" : 1 })

**> db.posts.update**({ title: 'Post One' },

... {

... $set: {

... comments: [

... {

... body: 'Comment One',

... user: 'Mary Williams',

... date: Date()

... },

... {

... body: 'Comment Two',

... user: 'Harry White',

... date: Date()

... }

... ]

... }

... })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.posts.find**({

... comments: {

... $elemMatch: {

... user: 'Mary Williams'

... }

... }

... }

... )

{ "\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"), "title" : "Post One", "body" : "Body of post one", "category" : "News", "tags" : [ "news", "events" ], "user" : { "name" : "John Doe", "status" : "author" }, "date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)", "comments" : [ { "body" : "Comment One", "user" : "Mary Williams", "date" : "Fri Dec 29 2023 14:16:07 GMT+0200 (EET)" }, { "body" : "Comment Two", "user" : "Harry White", "date" : "Fri Dec 29 2023 14:16:07 GMT+0200 (EET)" } ] }

**> db.posts.createIndex**({ title: 'text' })

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

**> db.posts.find**({

... $text: {

... $search: "\"Post O\""

... }

... })

{ "\_id" : ObjectId("658eb7df1a0ffa25d40ff2d1"), "title" : "Post One", "body" : "Body of post one", "category" : "News", "tags" : [ "news", "events" ], "user" : { "name" : "John Doe", "status" : "author" }, "date" : "Fri Dec 29 2023 14:13:19 GMT+0200 (EET)", "comments" : [ { "body" : "Comment One", "user" : "Mary Williams", "date" : "Fri Dec 29 2023 14:16:07 GMT+0200 (EET)" }, { "body" : "Comment Two", "user" : "Harry White", "date" : "Fri Dec 29 2023 14:16:07 GMT+0200 (EET)" } ] }

**> db.posts.find**({ views: { $gt: 2 } })

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"), "title" : "Post Two", "body" : "Body for post 2", "date" : "Fri Dec 29 2023 14:15:13 GMT+0200 (EET)", "category" : "Technology", "views" : 5 }

**> db.posts.**find({ views: { $gte: 7 } })

**> db.posts.find**({ views: { $lt: 7 } })

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"), "title" : "Post Two", "body" : "Body for post 2", "date" : "Fri Dec 29 2023 14:15:13 GMT+0200 (EET)", "category" : "Technology", "views" : 5 }

**> db.posts.find**({ views: { $lte: 7 } })

{ "\_id" : ObjectId("658eb7f21a0ffa25d40ff2d2"), "title" : "Post Two", "body" : "Body for post 2", "date" : "Fri Dec 29 2023 14:15:13 GMT+0200 (EET)", "category" : "Technology", "views" : 5 }

## Watching the “ExpressJS” video

In this course, I learnt how to use Express in Node, and built the Member Apps while coding along. I also learnt how to use uuid to generate universal UID, Express.Router to handle http request/response, and handlebars to create dynamic page in express.

Codes are here: <https://github.com/naosadhossen/expressJS>

## Watching the “Angular” video

In this course, l learnt how to install angular (npm I -g @angual/cli), initialize a angular project (ng new <project name>), and create components (ng generate component <component name>).

I also learnt how to start a ng server in a dev environment (ng serve) and build for production (ng build).

Also got myself familir with all the files and folders associated with angular project (components, app.module.ts, app.component.ts, package.json…).

Then coded along with the video.

Codes are here: <https://github.com/naosadhossen/angular>

## Watching the “MEAN-Stack” video

Finally, it is time to build our customized application based on the example project. I carefully watched the video and coded along with the video. The diary related to this part is in the following chapters.

Codes are here: https://github.com/naosadhossen/customerdb

**29.12.2023**

# Environment Setup

1. Git: Git was already installed in my laptop from previous course “Introduction to DevOps 2023-24”

naosadhossen@Naosad-Hossen customerdb % git -v

git version 2.43.0

1. Create GitHub repository for this project:

I have created an account in GitHub and a public repository by name “customerdb” with a gitignore file created for Node environment. Link to the repository: <https://github.com/naosadhossen/customerdb>

1. VS Code: VS Code was already installed in my laptop from previous course “Introduction to DevOps 2023-24”. Cloned the repository to VS Code.
2. Install MongoDB: Installed MongoDB version 3.6.23 and ran the MongoDB. Installed Mongo Compass version 1.40.4 and connected to the Database.

A screenshot of a computer

Description automatically generated

1. Install Node.JS: Downloaded node-v12.13.0.pkg and installed Node.js version 12 which is the recommended version in the webcourse.
2. Installed nodemon to automatically restart application after any change in the code. Command: npm install -g nodemon
3. Install Angular-CLI: Installed older version of angular-cli as recommended in the webcourse. Command: sudo npm install -g --unsafe-perm angular-cli
4. Install Flash messages package: Installed earlier version of Flash Messages package as recommend in the webcourse. Command: npm install angular2-flash-messages@1.0.8 --save

Now the environment is ready and proceeding to the coding phase.

# Running the example project

Now the idea is to run the example project in my development environment.

Create CustBE folder under the project folder CUSTOMERDB in VS Code.

CustBE: for backend code

A screenshot of a computer

Description automatically generated

Changed working direcroty to CustBE and initialized a project using npm init and populated the package.json with required dependencies. Then installed all the required packaged by running sudo npm install -g

Copied the back-end codes from the example project to CustBE with following modification in database string in config\database.js:

database: 'mongodb://@localhost:27017/customerdb'

Ran npm start to test the backend. Tested the api “/Register” from postman to register a new users, ‘/Authenticate” to login to the application, and “/profile” to fetch the user profile information.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

So all these three Backend API worked.

Now moving to frontend.

Changed the working directory to CustomerDB and ran ng new CustFE to create angular front end project. Changed the output directory in angular-cli.json as below to save the build files to /CustBE/Public folder:

"outDir": "../CustBE/Public"

Installed angular2-jwt manually:

npm install angular2-jwt @0.2.3 –save

To test the frontend, ran ng serve and it worked as I could see the login page.

A screenshot of a computer

Description automatically generated

Then ran ng build to generate the font-end pages. It generated the files and saved in it \CustBE\Public folder. Nodemon picked up the changes and could register, login and see the profile from Frontend.

Now it is time to push the code Front-end and Back-end Code to GitHub repository.

From CustomerDB directory execute:

Git add .

Git commit -m “example project-V0.5”

Git Push

**30.12.2023**

# Creating customized features for customer management

The idea is to create back-end api for:

1. Add a customer (/addcustomer)
2. Get all customer information (/getallcustomers)

Back-end API:

Created a file “customers.js” under /CustBE/models and defined customer schema as below:

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const config = require('../config/database');

// Customer Schema

const CustomerSchema = mongoose.Schema ({

name: {

type: String

},

email: {

type: String,

required: true

},

address: {

type: String,

required: true

},

phone: {

type: String,

required: true

},

country: {

type: String,

required: true

}

});

const Customer = module.exports = mongoose.model('Customer', CustomerSchema);

//Add Customer

module.exports.addCustomer = function(newCustomer, callback) {

newCustomer.save(callback);

}

//Find Customer

module.exports.geCustomerById = function(id, callback) {

Customer.findById(id, callback);

}

module.exports.getCustomerByUsername = function(name, callback) {

console.log(name);

const query = {name:name};

Customer.findOne(query, callback);

}

module.exports.getCustomerByEmail = function(email, callback) {

const query = {email:email};

Customer.findOne(query, callback);

}

module.exports.getCustomerByCountry = function(country, callback) {

const query = {country:country};

Customer.find(query, callback);

}

module.exports.getAllCustomer = function(callback){

Customer.find({},callback);

//console.log(result);

}

Created a new file “customers.js” under /CustBE/routes folder to create the api as below:

const express = require('express');

const customerrouter = express.Router();

const passport = require('passport');

const jwt = require('jsonwebtoken');

const config = require('../config/database');

const User = require('../models/user');

const Customer = require('../models/customers');

const { callbackify } = require('util');

const { resourceLimits } = require('worker\_threads');

// Testing Customers Router

customerrouter.get('/customer',(req, res, next)=>{

res.send("It is coming from Customer Route")

});

// Add a Customer

customerrouter.post('/addcustomer', passport.authenticate('jwt', {session:false}),(req, res, next) => {

let newCustomer = new Customer ({

name: req.body.name,

email: req.body.email,

address: req.body.address,

phone: req.body.phone,

country: req.body.country

});

Customer.addCustomer(newCustomer, (err, customer) => {

if(err) {

console.log(err);

res.json({success: false, msg: 'Failed to register user'});

} else {

res.json({success: true, msg: 'User registered'});

}

});

});

//Get All Customer Info

customerrouter.get('/getallcustomers', passport.authenticate('jwt', {session:false}), (req, res, next) => {

Customer.getAllCustomer((err,allCustomers)=>{

console.log(allCustomers);

res.send(allCustomers);

});

});

module.exports = customerrouter;

Added the customers routes in App.js as below:

const custrouter= require('./routes/customers');

app.use('/customers', custrouter); //customers routes

Tested the APIs from postman

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

So, both the API worked.

Now moving to CustFE to create front-end pages to add customer and view all customer information.

In the components folder, crated two components: addcust, custinfo

ng g component addcust

ng g component custinfo

Adding these two pages to navbar in navbar.component.html. These will be visible to users only after login.

<li class="nav-item" \*ngIf="authService.loggedIn()" [routerLinkActive]="['active']" [routerLinkActiveOptions] = "{exact:true}">

<a class="nav-link" [routerLink]="['/custinfo']">Customers <span></span></a>

</li>

<li class="nav-item" \*ngIf="authService.loggedIn()" [routerLinkActive]="['active']" [routerLinkActiveOptions] = "{exact:true}">

<a class="nav-link" [routerLink]="['/addcust']">Add a Customer <span></span></a>

</li>

Created a form in addcust.component.html with submit function onAddCustomerSubmit()

<div>

<h2 class="page-header">Add Your Customer</h2>

<form (submit)="onAddCustomerSubmit()" >

<div class="form-group">

<label for="name">Name</label>

<input type="text" [(ngModel)]="name" name="name" class="form-control" id="name" placeholder="Enter Name">

</div>

<div class="form-group">

<label for="email">Email</label>

<input type="email" [(ngModel)]="email" name="email" class="form-control" id="email" aria-describedby="emailHelp" placeholder="Enter Email">

<small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>

</div>

<div class="form-group">

<label for="address">Address</label>

<input type="text" [(ngModel)]="address" name="address" class="form-control" id="address" placeholder="Enter Address">

</div>

<div class="form-group">

<label for="phone">Phone Number</label>

<input type="text" [(ngModel)]="phone" name="phone" class="form-control" id="phone" placeholder="Enter Phone Number">

</div>

<div class="form-group">

<label for="country">Country</label>

<input type="text" [(ngModel)]="country" name="country" class="form-control" id="country" placeholder="Enter Country">

</div>

<input type="submit" class="btn btn-primary" value="Submit">

</form>

</div>

onAddCustomerSubmit() is defined in addcust.component.ts as below, which will validte the form input and call addCustomer() function from AuthService. If adding customer successful it will go to custinfo page, else it will go back to addcust page:

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

import { ValidateService } from '../../services/validate.service';

import { AuthService } from '../../services/auth.service';

import { FlashMessagesService } from 'angular2-flash-messages';

import { Router } from '@angular/router';

@Component({

selector: 'app-addcust',

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

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

})

export class AddcustComponent implements OnInit {

name: String;

email: String;

address: String;

phone: String;

country: String;

constructor(

private validateService: ValidateService,

private authService: AuthService,

private router: Router,

private flashMessage: FlashMessagesService) { }

ngOnInit() {

}

onAddCustomerSubmit(){

const customer = {

name: this.name,

email: this.email,

address: this.address,

phone: this.phone,

country: this.country

}

// Required Fields

if(!this.validateService.validateCustomerForm(customer)) {

this.flashMessage.show('Please fill in all fields', {cssClass: 'alert-danger', timeout: 3000});

return false;

}

//Add customer

this.authService.addCustomer(customer).subscribe(result=>{

if(result.success) {

this.flashMessage.show('Customer added successfully', {cssClass: 'alert-success', timeout: 3000});

this.router.navigate(['custinfo']);

} else {

this.flashMessage.show('Something went wrong', {cssClass: 'alert-danger', timeout: 3000});

this.router.navigate(['addcust']);

}

})

}

}

In AuthSerivce, addCustomer() function is defined as below which will call api 'customers/addcustomer':

customer: any;

addCustomer(customer) {

let headers = new Headers();

this.loadToken();

headers.append('Authorization', this.authToken);

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

return this.http.post('customers/addcustomer', customer, {headers: headers})

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

}

Now it is time to show all the customers information. Creating a table in custinfo.component.html to display all customers information.

<table class="table">

<thead class="thead-dark">

<tr>

<th scope="row">ID</th>

<th scope="row">Name</th>

<th scope="row">Email</th>

<th scope="row">Phone</th>

<!-- Add more table headers for other customer details if needed -->

</tr>

</thead>

<tbody>

<tr \*ngFor="let customer of customers">

<td>{{ customer.\_id }}</td>

<td>{{ customer.name }}</td>

<td>{{ customer.email }}</td>

<td>{{ customer.phone }}</td>

<!-- Display other customer details within additional table cells -->

</tr>

</tbody>

</table>

Calling the getCustomer() function in custinfo.component.ts when the page loads:

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

import { AuthService } from '../../services/auth.service';

import { Router } from '@angular/router';

@Component({

selector: 'app-custinfo',

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

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

})

export class CustinfoComponent implements OnInit {

customers:Object;

constructor(private authService:AuthService, private router:Router) { }

ngOnInit() {

this.authService.getCustomer().subscribe(data => {

this.customers=data;

//console.log(data);

},

err => {

console.log(err);

return false;

});

}

}

getCustomer() function is defined as below in authService.

getCustomer() {

let headers = new Headers();

this.loadToken();

headers.append('Authorization', this.authToken);

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

return this.http.get(‘/customers/getallcustomers', {headers: headers}).map(res => res.json());

}

Now processing to build the pages and test: ng build

form validation success:

A screenshot of a notebook

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

So, the example project code has been extened with the following features:

1. Backend API for to add customer in the customer database: /customers/addcustomer
2. Backend API for to get all customers information from database: /customers/getallcustomers
3. Frontend page for displaying a form to create a customer in the customer database: /addcust
4. Frontend page to display all customers in the customer database: /custinfo