Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

Timo Kivinen, 002351485

LEARNING DIARY, Full-Stack 2024 MODULE

LEARNING DIARY

10.9.2018

I checked the general information and understood the main focus of the course, which is to find my passion as a software developer and create a unique project to represent my skills. I chose frontend module because it was the most interesting project offered. I’ve also tried to set up my environment, but I could not decide which code editor I would like to use. I learned to set up a git repository and did my first commit, everything went smoothly after I clicked the banner to watch intro to GIT.

11.9.2018

I have chosen VS Code as my code editor for this course, I learned how to set up addons by googling how to do it. I searched the web for best addons and chose the best addons that I think fits me best. I started to watch the first part of the example project to understand the technologies better.

I did my second commit but somehow it did not go as I planned. I went to stackoverflow and found quite many threads about version control problems. I was able to figure out what was the problem and continued to watch the first part till the end.

14.3.2024

I learned about,

Something about Node.js (short video) and micro http server. Mostly it was refreshing my memory.

Setting up the environment.

16.3.2024

Watching NodeJS video and coding along.

17.3.2024

Studying Node.js video finished. Implementing later time (21.3.-22.3.2024).

Mongodb was installed and mongo database was created according to video.

Visual Studio code was integrated to github. Need some more studying.

18.3.2024

I wrote MongoDb exercises and test them. Here they are:

MONGODB

```
    Windows command prompt: mongosh
    test> show dbs
        acme    72.00 KiB
        admin   40.00 KiB
        config  72.00 KiB
        local   40.00 KiB
        test1   40.00 KiB
    test> use test1
        switched to db test1
    test1> show collections
        Products
    test1> db.dropDatabase()
        { ok: 1, dropped: 'test1' }
    test1> show dbs
        acme    72.00 KiB
        admin   40.00 KiB
        config  72.00 KiB
        local   40.00 KiB
    ..........................................
    test1> use acme
```

```
          switched to db acme
acme> show dbs
          acme    72.00 KiB
          admin   40.00 KiB
          config  72.00 KiB
          local   40.00 KiB
acme> db
          acme
acme> db.createCollection('posts')
          { ok: 1 }
acme> show collections
          posts
..........................................
db.posts.insert({title: 'Post One',
                              body: 'Body of post one',
                              category: 'News',
                              likes: 4,
                              tags: ['news', 'events'],
                              user: { name: 'John Doe',
                              status: 'author'          },
                              date: Date()})

..........................................
db.posts.insertMany([
          {
                    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 four',
                    category: 'Entertainment',
                    date: Date()
          }
          ])
..........................................
db.posts.find()
db.posts.find().pretty()

db.posts.find({category: 'News'})
db.posts.find.sort({title:1}).pretty()
db.posts.find().sort({title:1}).pretty()

db.posts.find({category: 'News'}).count()
db.posts.find().limit(2)

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

db.posts.find().forEach(function(doc) {print('Blog Post: ' + doc.title) })

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

db.posts.update({title: 'Post Two'},
        {
                title: 'Post Two',
                body: 'New post 2 body',
                date: Date()
        },
        {
                upsert: true
        }
)
===> ERROR

db.posts.update({title: 'Post Two'},
{
        $set: {
                body: 'Body of post 2',
                category: 'Technology'
        }
}
)

db.posts.find({title: 'Post Two'}).pretty()

db.posts.find({title: 'Post One'}).pretty()
..........................................

db.posts.update({title: 'Post One'}, {$rename: {likes: 'views'}})
db.posts.find({title: 'Post One'}).pretty()
..........................................
db.posts.remove({title: 'Post Four'})
db.posts.find()
..........................................

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

```
          $set: {
                    comments: [
                              {
                              user: 'Mary Williams',
                              body: 'Comment One',
                              date: Date()
                              },
                              {
                              user: 'Harry White',
                              body: 'Comment Two',
                              date: Date()
                              }

                    ]

          }
}
)
db.posts.find({title: 'Post One'}).pretty()
.........................................

db.posts.find({
comments: {
$elemMatch: {
          user: 'Mary Williams'
}
}
}
)

db.posts.createIndex({title: 'text'})
{
          "createdCollectionAutomatically": false,
          "numIndexesBefore": 1,
          "numIndexesAfter": 2,
          "ok": 1
}

db.posts.find({
          $text: {
                    $search: "\"Post O\""}})

.........................................

db.posts.update({title: 'Post Two'}, {$set: {views: 10}})
db.posts.find({views: {$gt:3}})
```

21.3.2024

There was problem with github so I created a new public repo https://github.com/tkivinen2024/NODE_CRASH_COURSE.

22.3.2024

Created os_demo.js, url_demo.js, event_demo.js, logger.js etc

Node Crash Course was implemented according Video examples.

Heroku client was installed and it was working with Visual Studio code. Heroku site was interesting my Credit card so i skip the last phase of the video and did’nt push codes to Heroku.

23.3.2024

The Day of repeating Mongodb examples.

Next thing is watching ExpressJS video and implementing the examples.

This is the second time to implement these Express exercises. I think it’s a good thing repeat implementation techniques: at a later time it’s easy to remember many technical details (and discuss with technical persons).

26.3.2024

Starting Angular course.

29.3.2024

Continue to implement Angular course (from 30 minutes in video). Plenty of technical problems because some coding errors. Anyway, it was nice 10 hours journey to Angular secrets.

30.3.2024

Another 10 hours course day. Angular implementation is now ok except one issue.

When new task is inserted, error occurs if you try toggle newly created task. The page must be refreshed.

31.3.2024

Starting Mean Stack part.

The first thing: MEAN Stack Front To Back [Part 1] - Project Introduction

MEAN Stack Front To Back [Part 2] - Express Setup & Routes

1.4.2024

Finished course video: “MEAN Stack Front To Back [Part 2] - Express Setup & Routes”.

I changed packagfe.json a little bit: Mongoose and node must be compatible:

"mongoose": "^5.13.22". That seems to be compatible with node 12.22.12.

3.4.2024

Finished course video: “MEAN Stack Front To Back [Part 3] - User Model & Register”.

There are some issues in the code. Http Post throws an exception.

4.4.2024

There is a fix for previous mentioned error. Http headers must contain Content-Type AND Content-Length when Postman is being used.

There was a total mess with my Github but now it is allright.

Here is a good advice when you have problems with Visual Studio project:

> *rm node_modules*
>
> *rm package-lock.json*
>
> *npm cache clean --force*
>
> *Package.json file:*
>
> > *"mongoose": "^5.13.22",*
>
> *npm install*
>
> *npm install -g nodemon*
>
> > *Step 1 : Go to this location --> File C:\Users\timoj\AppData\Roaming\npm*
> >
> > *Step 2 : Delete the nodemon.ps1 file and run the command.*
>
> *Nodemon*

Authentication method didn’t work correctly.

Must change from

```
const token = jwt.sign(user, config.secret, {expiresIn: 604800});
```

to

```
const token = jwt.sign({user}, config.secret, {expiresIn: 604800});
```

Thanks StackOverflow again.

....

5.4-6.4 .2024

It was time to implement "MEAN Stack Front To Back [Part 5] - Angular 2 Components & Routes" part of the course.

There were only few problems which was a surprise.

Some installation tips (old Angular-cli must be installed):

```
npm uninstall -g @angular/cli
npm install -g  @angular/cli@1.7.0
```

In components folder:

```
ng g component navbar
ng g component login
ng g component register
ng g component home
ng g component dashboard
ng g component profile
```

Index.html href must be an older version:

```
<link rel="stylesheet" href = "https://bootswatch.com/3/sandstone/bootstrap.min.css">
```

8.4-9.4 .2024

Started to learn Part 6 "Register Component, Validation & Flash Messages". There were examples how to register a component and create a submit button.

After them it was time to create validation service. New service had to add manually in to app.modules.ts. I couldn’t find exact the same example for validateEmail than in course video but found similar function from Stackoverflow forum.
My validateRegister function behaves opposite way than in course video but it seems to work correctly.
There was something errors after installing angular2-flash but after uninstall and reinstall it was running ok. I didn’t run audit fix.
My application works fine at the moment.

10.4-11.4 .2024
Two days long development cycle (full of technical issues, Part 7-Auth Service & User Registration).
Function called “registerUser” was outdated and there were in Video comment section several proposals to change the code. It took plenty of time because none of them didn’t work right away.
At the end of this period the following issue arises: “login sessions require session support”.
I reinstalled mongoose and passport:
Original:
"mongoose": "^5.13.22"
"passport": "^0.7.0"
New:
npm install passport@0.4.0 passport-local@1.0.0 passport-local-mongoose@5.0.1 --save
Now starts Video Part 8 – Login and logout…

12.4.2024

```
  authenticateUser(user: { username: String; password: String; }) {
    const httpOptions = {
      headers: new HttpHeaders({
        'Content-Type':  'application/json'
      })
    };
    return this.http.post<any>('http://localhost:3000/users/authenticate',
user, httpOptions);
  }
```