YARGS PART – I

const chalk = require('chalk');

const yargs = require('yargs');

const command = process.argv[2];

//Yargs ile birlikte gelen 2 tane komut vardir bunlar

//--help ve --version

//--version u kendimize gore asagidaki gibi ozellestirebiliriz.

//Customize Yargs Version

yargs.version('1.1.0');

// Create Add Command

yargs.command({ //Bu yargs modulunun cagiracagimiz bir opsiyonudur.

    command: 'add', //Ve burada ne yapacagimizi tanimlariz. add yerine istediginizide yazabilirsiniz.

    describe: 'Add a new note', //Yazmak zorunda degiliz ama yazmak sonradan help ile bilgi alimi icin faydali olacak.

    builder:{

        title: {

            describe: 'Note title', //Ve buradaki title, --title="Shopping List" deki title ile ayni titledir.

            demandOption: true, //Bunu true yaparak --title="Shopping List" i zorunlu yazmayi mecbur birakiyoruz. Sadece --title= yazarsak bize boelan deger yani true donderir.

            type: 'string' // Bu sekildede --title=15 yazarsak 15i bir string olarak kaydeder. Bunu atamadan 15 yazarsak veya number yazarsak numara olarak kabul eder.

        }

    },

    handler: (adamArgv)=>{ //Ve artik node app.js add yazdiktan sonra --title="Shopping List" yazarsak yeni bir komut eklemis olacagiz. Ve bize verecegi cikti su olacaktir. //Adding a new note { \_: [ 'add' ], title: 'Shopping List', '$0': 'app.js' }

        console.log('Adding a new note', adamArgv);

    }

});

// Bu yukaridaki koddan daha sonra

// node app.js --help yazarsak yargsa Commands basligi altinda "app.js / add /  Add a new note" bunlari ekledigimizi gorecegiz.

// node app.js add yazarsak handler daki fonksiyonumuz calisacak.

//Create Another Add Commands

yargs.command({

    command: 'anotherAdd',

    describe: 'Another add a note',

    builder:{

        title: { //Cagirirken consoledan muhtlaka basina -- KOY

            describe: 'Note title',

            demandOption: true,

            type: 'string'

        },

        body: {

            describe: 'body icin yap',

            demandOption: true,

            type: 'string'

        },

        denemeNumber: {

            describe: 'Deneme bir sayi gir',

            demandOption: false,

            type: 'number'

        }

    },

    handler: (anotherArgv)=>{

        console.log('Title : ' + anotherArgv.title );

        console.log('Body : ' + anotherArgv.body );

    }

});

// Bunu cagirmak icin = node app.js anotherAdd --title="This is a title" --body="This is a body" yazmaliyiz.

//CIKTI

//Title : This is a title

//Body : This is a body

//Create Remove Commands

yargs.command({

    command: 'remove',

    describe: 'Remove a note',

    handler: ()=>{

        console.log('Removing the note');

    }

});

//CHALLENNGE

//Create List Command

yargs.command({

    command: 'list',

    describe: 'Placeholder message for listing',

    handler: ()=>{

        let Listing = 'Listing';

        console.log(Listing);

    }

});

//Create Read Command

yargs.command({

    command: 'read',

    describe: 'Placeholde for reading',

    handler: ()=>{

        let Reading = 'Reading is good';

        console.log(Reading);

    }

});

//MY CHALLENGE

yargs.command({

    command: "update",

    describe: "Updating a note",

    builder: {

        title: {

            describe: "Title for update",

            demandOption: true,

            type: "string"

        },

        number: {

            describe: "Number of queue",

            demandOption: true,

            type: "number"

        }

    },

    handler: (builder)=>{

        console.log("Your title that is on the queue " + builder.number + " which is name is " + builder.title + " updated");

    }

});

// add, remove, read, list

// console.log(yargs.argv);

//Bu komutu iptal edersek yukaridaki hicbir kod calismaz. Bu kodla birlikte yukaridaki herbir kodu calistirdigimizda bir bu kodu birde cagirdigimiz kodu cikti aliriz.

//Yada asagidaki kodu kullanip sadece yukarida cagirdigimzi kodlari cikti alabiliriz.

yargs.parse();

**FILE SYSTEM MODULE**

const fs = require('fs');

const book = {

    title: 'Ego is the Enemy',

    author: 'Ryan Holiday'

}

/\*

//Burada yukaridaki book objemizi duz yazi haline ceviriyoruz.

//JSON.stringify bu ise yariyor.

const bookJSON = JSON.stringify(book);

console.log(bookJSON);

//Burada ise stringify yaptigimiz yazimizi tekrar eski hali objeye ceviriyoruz.

//Strigfy da iken title author gibi degerlerine erisemezken simdi erisebiliriz tekrardan.

const parsedDate = JSON.parse(bookJSON);

console.log(parsedDate.author);

\*/

const bookJSON = JSON.stringify(book);

// fs.writeFileSync('1-json.json', bookJSON); //Bu kod bize bu yazilarimizin oldugu simdiki dosya ile ayni satirda "1-json.json" isminde bir dosya olusturacak ve 2. olarak 2. parametredeki yazdigimiz seyi onun icine koyacak.

const dateBuffer = fs.readFileSync('1-json.json'); //readFileSync ilede dosyalarimizin iclerindeki verileri okuyabiliyoruz. Tek degisken aliyor oda okumak istedigimiz dosya yolunu buradaki gibi yaziyoruz.

console.log(dateBuffer); // Eger bu sekilde cikti alirsak bit ve bytes lar bize donecektir. Dogru cikti icin .toString() eklememiz gerekmektedir.

console.log(dateBuffer.toString());

const dataJSON = dateBuffer.toString(); //Ve burada okumus oldugumuz dosyayi stringe yani duz yaziya ceviriyoruz.

const data = JSON.parse(dataJSON); //Buradada parse ederek tekrar obje haline ceviriyoruz.

console.log(data.author); //Ve artik buradaki gibi data objemizin istedigimiz her elemanina ulasabiliriz.

//CHALLENGE TIME

const dataStringify = fs.readFileSync('1-json.json');

const dataStringifyBuffer = dataStringify.toString();

let data = JSON.parse(dataStringifyBuffer);

data.name = 'Mucahid',

data.planet = 'Mars',

data.age = 28

const dataStringify2 = JSON.stringify(data);

fs.writeFileSync('1-json.json', dataStringify2);

console.log(dataStringify2);

BCRYPTJS

const bcrypt = require('bcryptjs');

const myFunction = async () =>{

    const password = '1234567';

    const hashedPassword = await bcrypt.hash(password, 8)

//bcrypt.hash lemesini isteriz ve icine ilk olarak neyi hashlemesini istedigimizi yazariz mesela ornekte password yazilmis.

//Ikinci degere ise tur sayisini gireriz kac kere hash turu yapilmasini istedigimizi.

//En hizlisi ve guvenlisi 8 ama siz yinede istediginizide yapabilirsiniz. Ama yinede cok az yada cok ucuk fazla kullanmayin.

//Cok az kullanirsaniz hashiniz kirilabilir cok fazla kullanirsaniz uygulamanizi cokertebilir veya yavaslatabilirsiniz.

//Bu 8 uygulamayi yaratan kisilerin onerdigi rakamdir.

    console.log(password);

    console.log(hashedPassword);

//Peki geri dondurulemez bu verileri biz giris yaparken nasil kullanacagiz. Bcrypt bize burda bir guzellik sagliyor. Karsilastirma yapabilmemiz icin bir metodumuz var.

//Once bir degisken atayalim. Bunu konsolda gostermek icin yapiyorum bu degiskeni.

//Bu bize eger eslesirse true eslesmezse false donecek

//bcrypt yazip indirdigimiz ve daha sorna tanimladigimiz module erisip compare fonksiyonuna erisiyoruz.

//Daha sonra ilk fonksiyon degiskenine paralamizi ikincisine ise karsilastirmak istedigimiz hashlenmis parolayi yaziyoruz. Bu bize TRUE deger dondurecektir ayni oldugu icin.

    const isMatch = await bcrypt.compare('1234567', hashedPassword)

    console.log(isMatch);

}

myFunction()

\*Encryption algoritmalar ile hashing algoritmalar arasinda cok onemli bir fark vardir. Encryption algoritmalari istedigimiz zaten geri donderip neyi encrypt ettiysek onu geri alabiliriz. Fakat Hashing algoritmalarda olusturdugumuz hashleri bir daha asla geri donderemeyiz. Buda kirilmasini cok zorlastiriyor hashlenmis verilerin.

JSONWEBTOKEN

const jwt = require('jsonwebtoken');

//Ayni passportJS gibi authentication islemlerimizi kolaylastiran bi modul yapisidir.

//Asagida manuel statik olarak bir id verdim bunu daha sorna dinamik oalrak verecegiz

//jwt.sign fonksiyonunuz ikinci parametresine ise herhangi birsey yazabiliriz.

//console.log ile tokeni yazdirdigimizda jwt modulunun bize verdigi ozel bir hashlenmis bir tokenimiz olacak elimizde.

//eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiJhYmMxMjMiLCJpYXQiOjE1NzY0MTM1MzR9.NYBRxXKaXLTsB-ochX0eXJeKKEdhaKG8b7hIwNVroGI

//Bu token yukaridaki gibi noktalarla ayrilmis 3 parca bir tokendir

//1. parca meta bilgileri iceriyor. Hashlemede kullandigi algoritma ve JWT gibi. Asagidaki gibi

// {"alg":"HS256","typ":"JWT"}

//2. parca bir json stringidir. buda olayimizdaki id gibi bilgileri iceriyor. Asagidaki gibi

// {"\_id":"abc123","iat":1576413534} //ikinci deger zaman damgasi

//3. parca ise dogrulama tokenidir. Asagidaki gibi

// Y@=xz8+9\_dOlc>yEY;

const myFunction = async () =>{

    const token = jwt.sign({\_id: 'abc123'}, 'thisismynewcourse', {expiresIn: '7 days'}); //expiresIn ile tokenimizin ne kadar sure aktif kalacagini ayarlayabiliyoruz.

    console.log(token);

    //Dogrulama yapmak

    //Bu sekilde dogrulama yapariz vebunu degiskene atayip yazdirirsak asagidaki gibi bir cikti aliriz.

    //{ \_id: 'abc123', iat: 1576414301 }

    //Eger ikinci yazdigimiz yazi yukarida tokeni uyusturdugumzu yazi ile ayni olmasaydi imzamiz dogrulanmayacagindan oturu 'invalid signature' diye uzun bir hata alirdik.

    const data = jwt.verify(token, 'thisismynewcourse');

    console.log(data);

}

myFunction()

//Ornek Authentication

//Login Postumuz

router.post('/users/login', async(req,res)=>{

    try {

        const user = await User.findByCredentials(req.body.email, req.body.password)

        const token = await user.generateAuthToken()

//Burada gonderdigimiz user ve tokeni asagida postmandan login istegi ile post ettigimizde aldigimiz ciktiyi gorebilirsiniz.

        res.send({user, token});

    } catch (error){

        res.status(400).send()

    }

});

//UserSchemaya tanimladigimiz middleWare

//statics ler modellere erisebilir model metodlariyken, methodslar orneklere erisebilir ornek metodlar olarak isimlendirilirler.

userSchema.methods.generateAuthToken = async function(){

    const user = this

    const token = jwt.sign({\_id: user.id.toString()}, 'thisismynewcourse');

    return token

}

Daha sonra Postmandan Login istegine aldigimiz cevap

{

    "user": {

        "age": 0,

        "\_id": "5df622ffd3792352e0db4ab9",

        "name": "Mucahid Yazar",

        "email": "mucahidyazar@gmail.com",

        "password": "$2a$08$hlh9FoDuABncf036ozRrxuaKqstcqhLwAX9JBLCg8FRSSUaUUxo7e",

        "\_\_v": 0

    },

    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1ZGY2MjJmZmQzNzkyMzUyZTBkYjRhYjkiLCJpYXQiOjE1NzY0MTYyMzV9.27grjapd\_k2uXEutIkTo7IdAKOjNf31aJ1bJRwAQJ2g"

}

Tabi bunun daha bir kisayoluda bunu middleWare olusturarak yapmaktir.

MULTER

const upload = multer({

    //dest: 'avatars',

    limits: {

        fileSize: 1000000

    },

    fileFilter(req, file, cb){

        if(!file.originalname.match(/\.(jpg|jpeg|png)$/)){

            return cb(new Error('Please upload an image'));

        }

        cb(undefined, true);

    }

});

router.post('/users/me/avatar', auth, upload.single('avatar'), async (req, res)=>{

    req.user.avatar = req.file.buffer;

    await req.user.save();

    res.send(req.user);

},

//Asagidaki gibi ayarlarsak error message kismini karisik uzun bir error mesaji yerine huzel bir json objesi seklinde sadece error un yazdigi bir error mesage si alabiliriz

(error, req, res, next)=>{

    res.status(400).send({ error: error.message });

});

router.delete('/users/me/avatar', auth, async (req,res)=>{

    req.user.avatar = undefined;

    req.user.save();

    res.send(req.user);

});

//Buffer i res.set ile contenttype image/jpg olacak sekilde ayarliyoruz.

//ayarladiktan daha sonra user.avatar i send yaparak resmi gosteriyoruz.

router.get('/users/:id/avatar', async(req, res)=>{

    try{

        const user = await User.findById(req.params.id);

        if(!user || !user.avatar){

            throw new Error();

        }

        res.set('Content-Type', 'image/jpg');

        res.send(user.avatar)

    } catch(error){

        res.status(404).send();

    }

})

SENGRID KULLANIMI

const sgMail = require('@sendgrid/mail');

const sendgripAPIKey = 'SG.heRHIk6FT6uMhP4f\_-a86w.pIt46aYKbQb-63GPE1lIlmV4IRfcFPTazEBHRYBZ7JU';

sgMail.setApiKey(sendgripAPIKey);

sgMail.send({

    to: 'mucahidyazar@gmail.com',

    from: 'admin@mucahidyazar.com',

    subject: 'This is my first creation',

    text: 'I hope this one actually get to you'

});

ENV-CMD

npm i env-cmd

Ilk once yukaridaki kod ile yukuyoruz.

Daha sonra ana dizine config ve icine dev.env dosyasini olusturuyoruz.

Ve icine asagidaki gibi saklamak istedigimiz verileri giriyoruz.

PORT=3000

SENDGRID\_API\_KEY=SG.heRHIk6FT6uMhP4f\_-a86w.pIt46aYKbQb-63GPE1lIlmV4IRfcFPTazEBHRYBZ7JU

Ve son olarak asagidaki gibi packagejsonda olan script komutumuzu calistirip sunucumuzu baslatiyoruz.

"dev": "env-cmd -f ./config/dev.env nodemon src/index.js"

HEROKU ENVORIEMENT KEY AYARLAMAK

heroku config:set key=value

Yukaridaki kod seklinde ayarlamalarimizi konsola yazarak yapiyoruz envioriment degiskenleri icin.

Heroku config

Yukuaridaki kod ilede daha once ayarladiginiz keyleri gorebilirsiniz.

heroku config:unset key

Yukaridaki kodlada istediginiz heroku enviroment keyi kaldirabilirsiniz.

Git push heroku master diyerek en son push lamayi unutmayin master branchinizi.

JESTJS.IO

Test dosyalari yazip projemizi kontrol edip sorunlari hizlica tespit etmeye yarar.

Bir kere ayarladiktan sonra bir daha test yazmaniza gerek kalmayacak.

Once bir tests klasoru olusturup icine math.test.js dosyasi olusturuyoruz. Test dosyalarimizin uzantilari .test.js ile bitmek zorunda cunku gormesi ve test dosyasi oldugunu anlamasi icin boyle bitmek zorunda.

Ve package.json dosyasinada script olarak “test”: “jest” olarak kayitli olmali ve bunu testlerimizi kontrol etmek istedigimizde npm test yazarak calistirmaliyiz.

test('Hellow world!', ()=>{

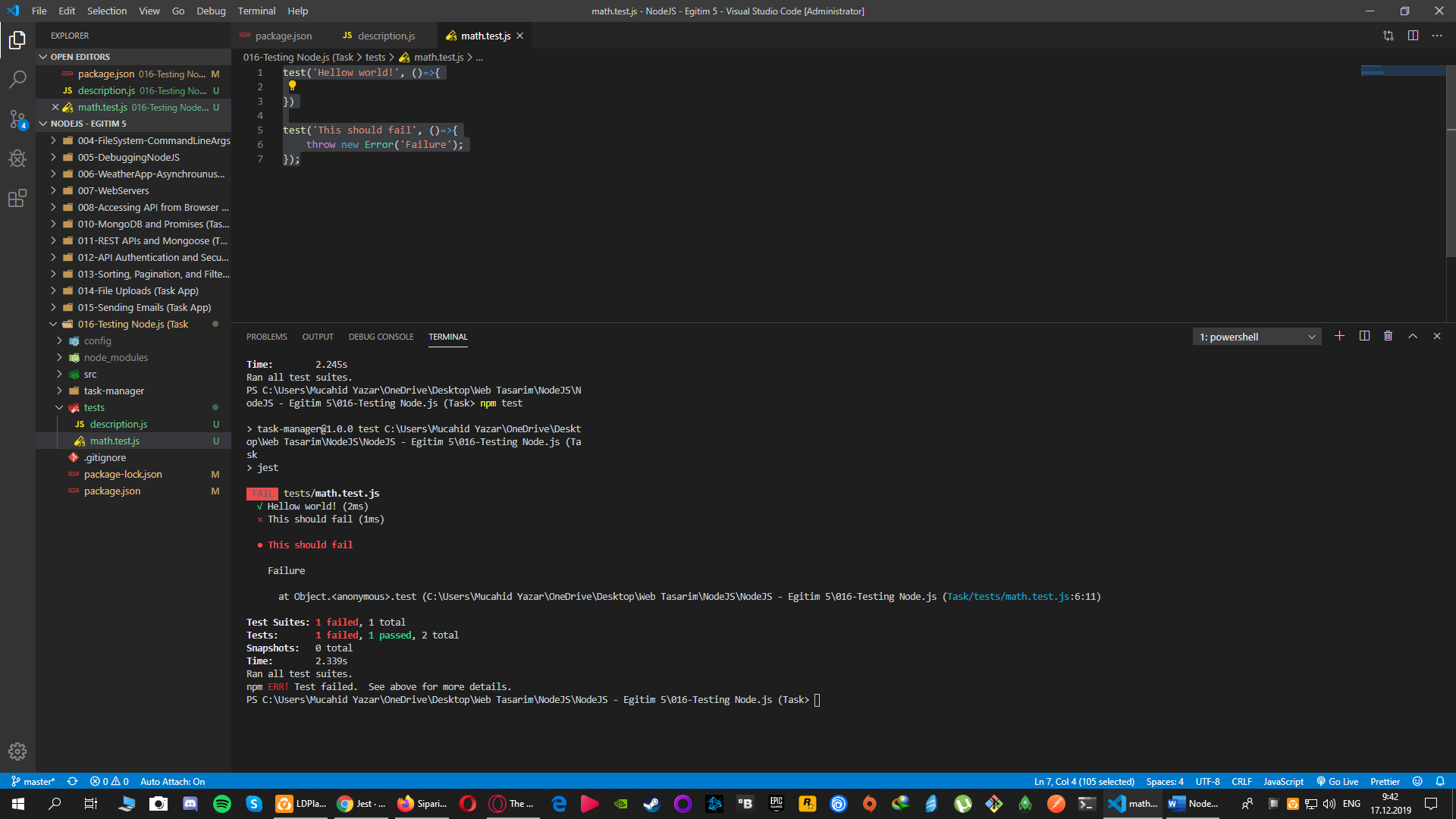
})

test('This should fail', ()=>{

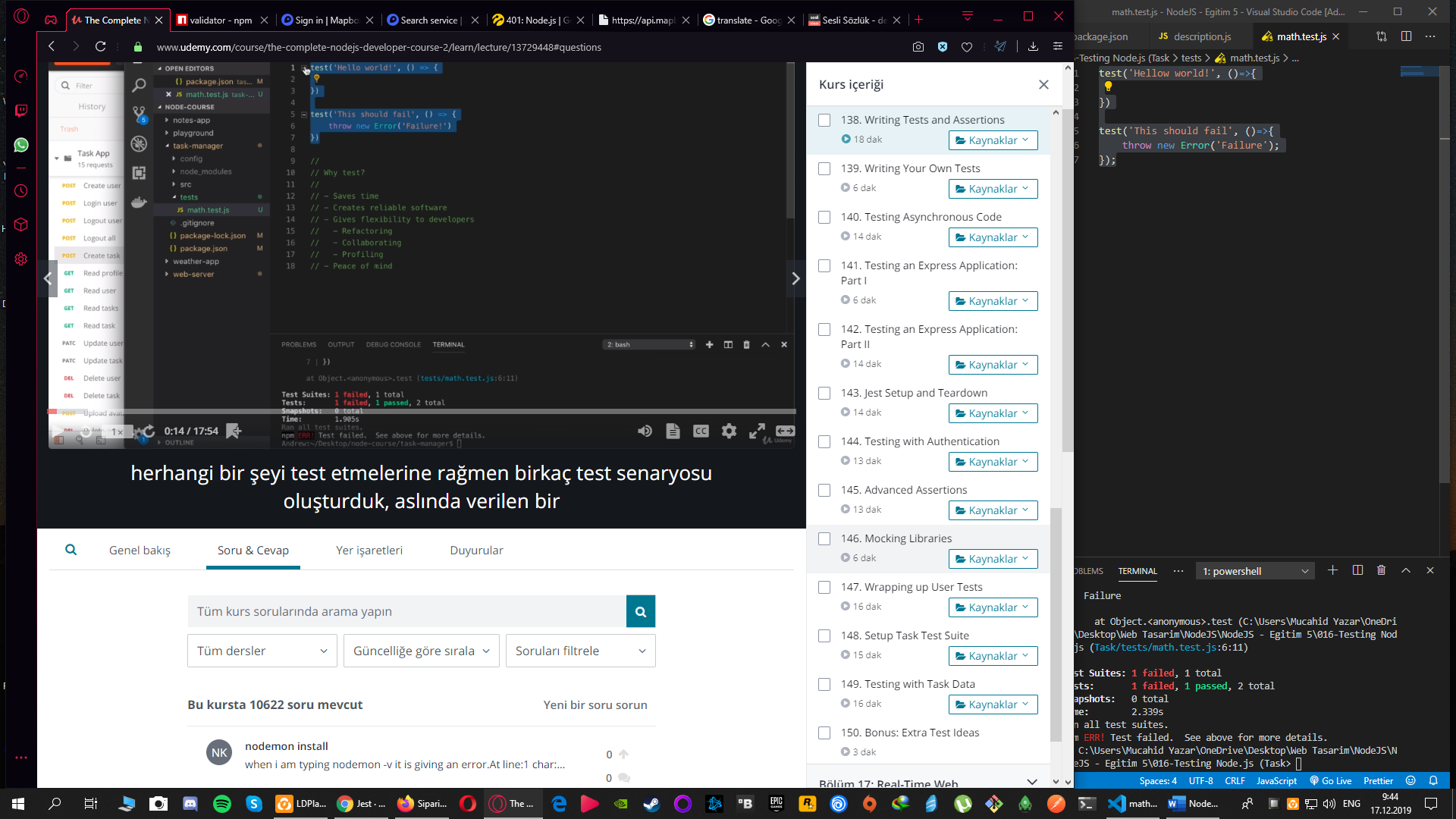
    throw new Error('Failure');

});

Yukaridaki ornek test kodunun ciktisi asagidaki gibidir.



Neden TEST?



Package.json da test’te jext yanina --watch yazarak testimizin surekli kontrol edip izlemesini sagliyoruz.

Ve bu bize npm test yazip calistirdigimizda asagidaki gibi bir menu gosterir.

W ile watch menusunu yeniden acabiliriz terminalde kaybolursa.

Yani bu sayede testi surekli surekli bastan calistirmamiza gerek kalmiyor. Bir nevi testin nodemonu gibi.

