**Hướng dẫn: express, mongoose, morgan, cookie-parse, join, wager, http-status-code, bcryptjs, jsonwebtoken**

***Tạo hệ thống thư mục backend và cài đặt nodemon khi “npm start”***

***Tạo Server nodes với Express, Mongoose, Morgan, cookie-parse, Nodemon***

***Nạp thư viện Mongoose, cookie-parse và morgan***

***Khai báo bodyParser trong server (từ express 4.0 bodyParser được đưa vào express)***

***Tạo file model: userModels.js***

***Tạo đăng ký người dùng: userController -> authRoutes ->*** [***http://.../appchat/register***](http://.../appchat/register)

***Validation data input: Joi***

***Nạp thư viện http-status-codes: hỗ trợ thông tin lỗi kiểm về res.status(int lỗi) bởi HttpStatus***

***Noi dung CreateUser:***

***1 kiem tra req.body bằng joi các giá trị hợp lệ - và tạo Helper.js để tạo function hỗ trợ***

***2 Nếu hợp lệ thì tìm email và username trong csdl – Nếu có thì báo có tồn tại và thoát***

***3 Nếu không thì bcrypt password thành hash***

***4 Tạo jwt của jsonwebtoken và trả cookie jwt về cho front-end***

***4 Nạp vào database và thông báo thành công hay thất bại***

***Phần Font-end:***

***Tạo services/auth dùng promise post thông tin account qua back-end dùng HttpClientModule***

***Sử dụng FormGroup, FormBuilder, Validators: tạo biến form, form.Formbuilder.Group truy xuất giá trị biến, Validator để kiểm tra điều kiện form. Trình duyệt báo lỗi [DOM] Input elements should have autocomplete attributes: khắc phục thêm input autocomplete="Off"***

***Thực hiện hiển thị lỗi lên trình duyệt***

***1. Khóa nút đăng ký khi có lỗi***

***2. Hiện lỗi khi chưa nhập***

***3. Hiện lỗi trả về từ backend***

***Tạo component và module stream và module stream.routing: khi đăng ký tự động chuyển về Streams***

***Tạo spinner: class=”progress” class=”indeterminate” của materialcss***

***Phần Backend:***

***Tạo phần login: Nếu ten và pass trống thì lỗi, có thì kiểm tra tồn tại, nếu có thì kiểm tra pass thông qua bcrypt.compare, nếu sai thì báo sai pass, đúng thì trả về token***

1. Tạo server nodejs với các thư viện:

Npm init

Npm i --save express mongoose morgan

Npm i -g Nodemon

2. Tạo hệ thống thư mục backend theo mvc

Trong package.json thêm: “start”: nodemon server

\app\models – controllers – routes – Helpers – Config

3. Nạp thư viện: Mongoose, cookie-parser, morgan

Const mongoose = require(‘mongoose’) và mongoose.Promise = global.Promise và mongoose.connect();

Const cookieParser = require(‘cookie-parser’) và app.use(cookieParser());

Const logger = require(‘morgan’) và app.use(logger(‘dev’));

4. Nạp bodyParser xử lý địa chỉ và dữ liệu từ browser lên server (trong express và khác cookie-parser)

app.use(express.json({ limit: "50mb" }));

app.use(express.urlencoded({ extended: true, limit: "50mb" }));

5. Tạo userModel.js và Tạo Đăng ký người dùng

UserModel { username, email, password }

Controller/authController: CreateUser

Routes/authRoutes: POST api/auth/register 🡪 authController:CreateUser

6. Dùng validation Joi cho kiểm tra dữ liệu đầu vào cho Nodejs

Npm i --save Joi

Joi: tạo 1 biến tạm schema:Joi.object().keys() chứa các điều kiện cho biến tạm schema

C1-- Dùng Joi.validation (nơi lấy, schema, (result, error) => {})

C2-- Dùng const {err, result} = Joi.validation(nơi lấy, schema)

7. Dùng http-status-codes để chuyển thông báo lỗi vào hệ thống thông qua biến http-status-codes

Npm i http-status-codes –save và constant HttpStatus = require(‘http-status-codes’)

Và res.status (HttpStatus.BAD\_REQUEST)

8. Dùng async/await với 1 trong 2 loại: callback hay function để kiểm tra email và username

9. Dùng bcryptjs để mã hóa pass

Npm i --save bcryptjs

return bcrypt.hash(biến password, 10, (err, hash) => {}

10. Dùng jsonwebtoken để tạo biến token = jwt.sign(obj, keysecret, interval)

11. Tạo trên database và save

User.create(account).then(user) => {

Res.status(HttpStatus.CREATED).json({messenger: “Success”});

FRONT END:

12. Tạo service để nhận Observerbel để nhận subcrcibe (biến) từ from nhập liệu

Phần backend dùng port 3000 và frontend dùng port 4200 nên cần dùng CORS

13. Form cần dùng 3 thư viện FormsModule: FormGroup, FormBuilder, ReactiveFormsModule:Validatior từ @angular/form

FormGroup : tạo biến nhập [formGroup]=”biến”

FormBuilder : để lấy biến đơn từ biến form kiểm tra validator : formControlName=”bien\_”

Validator : để kiểm tra biến formControlName

Kiểm tra lỗi từ form: <span class=”error” \*ngIf=”!biến.controls[‘bien\_’].valid && biến.controls[‘bien\_’].touched”> Thông báo lỗi </span>

14. Thực hiện hiển thị lỗi lên trình duyệt gồm 3 phần:

1. Khóa nút đăng ký khi có lỗi: [disable]=!signupForm.valid

2. Hiện lỗi khi chưa nhập: class="error" \*ngIf="!signupForm.controls['email'].valid && signupForm.controls['email'].touched"

3. Hiện lỗi trả về từ backend: if (err.error.msg) – if(err.error.message) thì div id="errorMsg" \*ngIf="errorMessage"

15. Tạo component và module stream và module stream.routing: khi đăng ký tự động chuyển về Streams

Tạo components/Streams/

Tạo module/streams.module

Tạo moduel/stream.routing.module

App.module gọi Stream.module và Stream.routing.module

Stream.module gọi StreamComponent

StreamComponent gọi thư viện Router @angular/router(thuộc RouterModule) : nhiệm vụ route đến component theo cú pháp: this.router.navigator([‘component’]);

Phần Back-end:

16. Viết phần login:

!Req.body.usename || !Req.body.password thì INTERNAL\_SERVER\_ERROR

User.findOne then (user) if (!user) thì NOT\_FOUND

Có thì bcrypt.compare nếu !result thì INTERNAL\_SERVER\_ERROR

Const token = jwt.sign({data},secret,{expiresIn:1000} res.cookie(‘auth’,token) return res.status…